The Angband Manual

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Welcome to Angband

Angband is a complex single player dungeon simulation. A player (you!) creates a character, choosing from a variety of races and classes, and then plays that character over a period of days, weeks, even months.

The player will begin their adventure on the town level where they may acquire supplies, weapons, armor, and magical devices by buying from various shop owners. Then the player can descend into the Pits of Angband, where they will explore the many levels of the dungeon, gaining experience by killing fierce creatures, collecting powerful objects and valuable treasure, and returning to town occasionally to buy supplies. Eventually, as the player grows more experienced, they may attempt to win the game by defeating Morgoth, the Lord of Darkness, who resides far below the surface.

Angband is a very complex game, and it may be difficult to grasp everything at first, especially if you have never played a roguelike game before. A key resource for Angband players is the forum [http://angband.oook.cz], where you can ask for help, and also post compliments, complaints, suggestions, bug reports, and interesting experiences. Don't be shy!

Getting Started

The Manual

- Keymaps
- Colours
- Visuals
- Interface details

Meta

- Version Information
- Copying and licence information
- Credits

For Developers

- Modifying Angband
- Compiling Instructions
- Debug Command Descriptions
- How It Works
- Documentation on documentation

Indices and tables

- Index
- Module Index
- Search Page

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

A quick demonstration

Angband is a very complex game, so you may want to try the following quick demonstration. The following instructions are for demonstration purposes only, and so they are intentionally boring.

For this demo, we will assume that you have never played Angband before, that you have not requested any special “sub-windows”, that you have not requested any special “graphics” modes, that you have a “numeric keypad” on your computer, and that you are using the default options, including, in particular, the “original” command set. If any of these assumptions are incorrect, you will need to keep in mind that this demo may not work; in particular note that Windows users will be using graphical tiles by default, so unless you turn off graphics map symbols below will be replaced by little pictures. There are many ways to view this file while playing.

Any time you see the ‘-more-’ prompt, read the message and press space. This takes precedence over any other instructions. At any other prompt, for example, if you accidentally hit a key, you can normally “cancel” the action in progress by pressing escape.

When the game starts up, depending on what platform you are using, you may be taken directly to the character creation screen, or you may have to ask to create a new character by using the File menu. In either case, you will be shown the character information screen, and you will be given a series of choices. For this demo, press `a` three times to elect a “human warrior” character with the point-based stat allocation system. You will now be presented with a description of your character. Look over the description briefly, there is a lot of information here, and most of it will not make any

sense. Press enter four times and your character will be placed into the “town”.

You should now be looking at the basic dungeon interaction screen. To the left is some information about your character. To the right is an overhead view of the town. Nothing happens in Angband while the game is waiting for you to specify a command, so take a good look at the town. You will see a variety of symbols on the screen. Each symbol normally represents a terrain feature, an object, or a monster. The `@` symbol is special, it represents your character. You can use the `/` command to find out what a given symbol represents. Press “/” then `@` now to verify the meaning of the `@` symbol.

The solid blocks (which may be `#` symbols on some systems) around the edge of the town represent the walls that surround the town. You cannot leave the town above ground, although some games derived from Angband (called “variants”) have an overground element.

The 3x3 squares represent stores. The “numeric” symbols represent an “entrance” to a store. The `.` symbols represent the “floor”. It is currently daytime, so most of the town should consist of stores and illuminated floor grids. There will also be a few `:` symbols which represent “passable rubble”.

Any “alphabetic” symbols always represent monsters, where the word “monsters” specifies a wide variety of entities, including people, animals, plants, etc. Only a few “races” of monsters normally appear in town, and most of them are harmless (avoid any mercenaries or veterans if you see them). The most common “monsters” in town are small animals (cats and dogs) and townspeople (merchants, mercenaries, miscreants, etc).

Now use the `l` command to “look” around. This will cause the cursor to be moved onto each “interesting” square, one at a time, giving you a description of that square. The cursor always starts on the square containing your character. In this case, you will see a message telling you that your character is standing on a staircase. Keep pressing space until the prompt goes away.

Now press `i`, to display your character's "inventory". New characters start out with some objects to help them survive (though there is an option to start with more money instead). Your character will have some food, a potion, some torches, and a scroll. Press `e` to see what you are wearing. You will find you are wearing armour on your body, wielding a dagger and lighting the way with a torch. You have many other equipment slots but they are all currently empty.

Press `t` to take something off. Note that the equipment listing is reduced to those objects which can actually be taken off. Press `g` to take off the armour, and then press `e` again. Note that the armour is no longer shown in the equipment. Press escape. Press `w` to wield something and observe that the inventory listing is reduced to those objects which can actually be wielded or worn, press `e` to put the armour back on.

Monsters can only move after you use a command which takes "energy" from your character. So far, you have used the `w` and `t` commands, which take energy, and the `e`, `i`, `l`, and `/` commands, which are "free" commands, and so do not take any energy. In general, the only commands which take energy are the ones which require your character to perform some action in the world of the game, such as moving around, attacking monsters, and interacting with objects.

If there were any monsters near your character while you were experimenting with the `w` and `t` commands, you may have seen them "move" or even "attack" your character. Although unlikely, it is even possible that your character has already been killed. This is the only way to lose the game. So if you have already lost, simply exit the game and restart this demo.

One of the most important things that your character can do is move around. Use the numeric keys on the keypad to make your character move around. The `4`, `6`, `8`, and `2` keys move your character west, east, north, and south, and the `7`, `9`, `1` and `3` keys move your character diagonally. When your character first moves, observe the `>` symbol that is left behind. This is the "staircase" that she was standing on earlier in the demo - it is the entrance to the dungeon.

Attempting to stay away from monsters, try and move your character towards the entrance to the “general store”, which is represented as a `1` on the screen. As your character moves around, use the `l` command to look around. You can press escape at any time to cancel the looking. If you die, start over.

One of the hardest things for people to get used to, when playing games of this nature for the first time, is that the character is not the same as the player. The player presses keys, and looks at a computer screen, while the character performs complex actions, and interacts with a virtual world. The player decides what the character should do, and tells her to do it, and the character then performs the actions. These actions may induce some changes in the virtual world. Some of these changes may be apparent to the character, and information about the changes is then made available to the player by a variety of methods, including messages, character state changes, or visual changes to the screen. Some changes may only be apparent to the player.

There are also a whole set of things that the player can do that can not even be described in the virtual world inhabited by the character, such as resize windows, read online help files, modify colormaps, or change options. Some of these things may even affect the character in abstract ways, for example, the player can request that from now on all monsters know exactly where the character is at all times. Likewise, there are some things that the character does on a regular basis that the player may not even consider, such as digesting food, or searching for traps while walking down a hallway.

To make matters worse, as you get used to the difference between the player and the character, it becomes so “obvious” that you start to ignore it. At that point, you find yourself merging the player and the character in your mind, and you find yourself saying things like “So yesterday, I was at my friend’s house, and I stayed up late playing Angband, and I was attacked by some wild dogs, and I got killed by a demon, but I made it to the high score list”, in which the pronoun changes back and forth from the real world to the virtual one several times in the same sentence. So, from this point on you may have to separate the player and the character for yourself.

So anyway, keep walking towards the entrance to the general store until you actually walk into it. At this point, the screen should change to the store interaction screen. You will see the name of the shop-keeper, and the name of the shop, and a list of objects which are available. If there are more than twelve different objects, you can use the space or arrow keys to scroll the list of objects. The general store is the only store with a fixed inventory, although the amount of various items may vary. One of the items sold here are flasks of oil. Press 'down' to highlight the line with flasks of oil and press the `p` key to purchase some. If you are asked how many you want, just hit enter. Any time you are asked a question and there is already something under the cursor, pressing return will accept that choice. Hit enter to accept the price. Many commands work inside the store, for example, use the `i` command to see your inventory, with the new flask of oil. Note that your inventory is always kept sorted in a semi-logical order, so the indexes of some of the objects may change as your inventory changes.

Purchase a few more flasks of oil, if possible: this time, when asked how many you want, press `3` then return to buy three flasks at once. Flasks of oil are very important for low level characters, because not only can they be used to fuel a lantern (when you find one), but also they can be ignited and thrown at monsters from a distance. So it is often a good idea to have a few extra flasks of oil. Press escape to leave the store. If you want, take time to visit the rest of the stores. One of the buildings, marked with an `8`, is your "home", and is not a real store. You can drop things off at home and they will stay there until you return to pick them up. The interface is exactly the same as a store, but there is no payment.

Now move to the staircase, represented by the `>` symbol, and press `>`, to go down the stairs. At this point, you are in the dungeon. Use the `l` command to look around. Note that you are standing on a staircase leading back to town. Use the `<` command to take the stairs back to town. You may find that any townspeople that were here before have disappeared and new ones have appeared instead. Now use the `>` command to go back down the stairs into the dungeon. You are now in a different part of the dungeon than you were in before. The dungeon is so huge, once you leave one part of the dungeon, you will never find it again.

Now look at the screen. Your character may be in a lit room, represented as a large rectangle of illuminated floor grids (`.`), surrounded by walls. If you are not in a lit room, keep going back up to the town and back down into the dungeon until you are. Now look around. You may see some closed doors (+) or some open doors (`'`) or some open exits (`.`) in the walls which surround the room. If you do not, keep playing the stairway game until you are in such a room. This will keep the demo simple.

Now look around using the `l` command. You may see some monsters and/or some objects in the room with you. You may see some stairs up (<) or some stairs down (>). If you see any monsters, move up next to the monster, using the movement keys, and then try and move into the monster. This will cause you to attack the monster. Keep moving into the monster until you kill the monster, or it runs away, or you die. If you die, start a new game. If the monster runs away, ignore it, or chase it, but do not leave the room. Once all the monsters in the room are dead or gone, walk on top of any objects in the room. Press `g` to get the object, and it will be added to your inventory. If there are any closed doors (+) in the room walk up next to them and press `o` and then the direction key which would move you into the door, which should attempt to "open" the door.

Now use the movement keys to explore the dungeon. As you leave the room, you will probably notice that your character cannot see nearly as far as she could in the room. Also, you will notice that as she moves around, the screen keeps displaying some of the grids that your character has seen. Think of this as a kind of "map" superimposed on the world itself, the player can see the entire map, but the character can only see those parts of the world which are actually nearby. If the character gets near the edge of the "map" portion of the screen the entire map will scroll to show a new portion of the world. Only about ten percent of the dungeon level can be seen by the player at one time, but you can use the `L` command to look at other pieces of the map. Use the `.` key, then a direction, to "run" through the dungeon. Use the `R` key, then return, to force your character to "rest" until she has recovered from any damage she incurs while attacking monsters. Use the `M` key to see the entire dungeon level at once, and hit escape when

done. If your food rations are still at index `a` in your inventory, press `E`, `a` to eat some food. If your oil is still at index `b` in your inventory, and there is a monster nearby, press `v`, `b`, `'` to throw a flask of oil at the nearest monster. To drop an item from your inventory, press `d` plus the index of that item. You can use the ‘^X’ key to quit and save the game.

You now know enough to play a quick game of Angband. There is a lot more for you to learn, including how to interpret information about your character, how to create different kinds of characters, how to determine which equipment to wield/wear, how to use various kinds of objects, and how to use the more than fifty different commands available to your character. The best resource for learning these things is the online help, which include, among other things, a complete list of all commands available to you, and a list of all the symbols which you may encounter in the dungeon, and information about creating new characters.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

A Players' Guide to Angband

This guide assumes familiarity with the basic mechanics of the game. If you're completely new to Angband, check out the user's manual and just start playing, get into the dungeon and try, well, whatever seems to be prudent. You'll probably die rather quickly, but the following will make much more sense to you if you have just a little actual gameplay experience.

This guide was written for Angband 3.5.0, and is now a little out of date (although the worst of the obsolete stuff has been cleaned out). It is still a handy source of hints and advice; just don't take it as an authority.

The basics

As borrowed from a classic rgra post, there's not much that you actually have to do. Your one and only mission is to slay Morgoth on dungeon level 100. In order to get there, you need to go down a lot of stairs and kill Sauron on dungeon level 99. That's about it - everything else is optional. Of course, before you can kill Sauron you'll need lots of experience and good equipment, but by the time you get that deep you'll have both. Just see to it that you don't die along the way.

The one and most important thing you need to get in your head is that you can't possibly kill every monster on every level. Think of the game as forays into the dungeon from which you want to return with cool stuff; think of the shallower levels as obstacles you need to overcome on your way deeper into the dungeon; or whatever you

like. Just never-ever think of it as a killing spree. Until you find Sauron, your task is to survive and eventually get to level 99.

The next point is that you don't need to fight any particular monster (other than the big two). Yes, there might be a rather impressive hoard in that vault – but if the monsters guarding it are too many or too fierce, well, just give it a pass. Angband offers an endless supply of monsters and treasure and everything. There will always be another day – provided you live to see another day. Sometimes you just have to bug out and run for your life.

So, let me recap, the vital points of Angband are:

- go down a lot of stairs
- kill Sauron (dl 99)
- kill Morgoth (dl 100)
- Quick start

From this point, the guide assumes that you are playing a fighting class (warrior, ranger, rogue or paladin) and race (Dunedain, High-Elf, Dwarf, Half-Orc, Half-Troll.) First, use the 'cost-based' stats selection, and create a character with 3 or 4 blows (Don't pick a class/race combinations, like Dwarf Ranger, that can only get 2 blows.)

First, set your stats to get maximum blows. If you have a combination with good dexterity, select 18/10 dexterity, and then set your strength to 17 or 18. Set your spell-casting stat to something low but usable (~12), and spend the rest on constitution. For a beginning character, constitution is least important, because it doesn't add significant HP until well above 18.

Example character:

```
 [Angband 3.5.0 Character Dump]

Name   Anar          Age             102          Self  RE
Sex    Female        Height       5'11" STR: 17 +1 +0 +0
Race   High-Elf      Weight    14st 0lb   INT:     10  +3
Class  Ranger        Turns used:          WIS:     10  -1
Title  Runner        Game             1   DEX:     18  +3
```

```
HP      14/14          Standard          0   CON:      10  +1
SP      0/0            Resting           0

Level                    1    Armor         [0,+2]    Savir
Cur Exp                  0                             Steal
Max Exp                  0    Melee         1d4,+2    Disar
Adv Exp                 23    To-hit         23,+4    Magic
                              Blows       2.0/turn    Perce
Gold                   139                             Searc
Burden           18.5 lbs    Shoot to-dam      +0    Infra
Speed              Normal    To-hit         33,+4
Max Depth            Town    Shots         1/turn
```

A Fighting Chance

For a melee character, the most important measure of power is how much damage they can do in a single turn. Consider the above cl 1 character with 3 blows from a Rapier (+7, +9), in a fight vs an out-of-depth Bullroarer, with no escapes, no ranged weapons, and no armor. On the face, she has no chance. However, if she can get in the first blow, she in fact has a 71% chance of killing Bullroarer in a single turn, and a 95% chance of frightening him.

Further, with full buffing (!Hero, ?Blessing ?Berserk Strength), the character has an 81% chance of killing him in a single turn. As well as improving the to-hit probability from 91% to 95%, this corresponds to a 50% reduction in the chance of failure in a dangerous situation. Finally, assuming ‘Anar’ does win this battle, she gets 450 experience, and immediately goes jumps to cl 8.

Missile Damage

With proper preparation even a weak character has an almost guaranteed chance of killing Bullroarer, if you meet him at a distance (across a lighted room). An unenchanted longbow does 7.5 damage/shot with ordinary (unenchanted) arrows. However, flasks of oil do 7HP nominal damage when thrown, and do triple damage (21HP) vs fire-vulnerable monsters. Bullroarer has 60 HP, so even a small stack of oil will finish him off.

Summary

- Character power is more closely associated with damage output rather than HP or character level.
- If you preserve your supplies, you can fight well above your weight. Early in the game, this means using flasks of oil against worthwhile targets; later in the game this means using branded or slaying “ego” ammunition. Good ammunition is too valuable to waste on less valuable targets (like red jellies, or groups of orcs.)
- Buffing, generally with !Heroism, can be very helpful to get starting characters out of sticky situations.
- Going deeper in the dungeon is often a more conservative (safer) strategy than staying at a shallow (cl < dl) depth. (cl: character level; dl: dungeon level) HP and character level are easy to come by.
- Starting Equipment

Fighting Power

Buy a light weapon that gives you the maximum possible blows, with the highest dice available. This is generally a Rapier, Main Gauche, or Dagger. If these are not available, you may be better off quitting and restarting.

Plan to be deep

Always make sure to have a scroll of recall, even on your first trip into the dungeon. Monsters and items at dl 5 (250’) and deeper are much (~100x) more valuable than items at dl 1. (The example above may be contrived, but it is representative.)

To survive deep(er) you will want:

a. Escapes:

    - 3+ ?Phase Door
    - 1 ?Recall

a. Protection from secondary effects (confusion, blindness, poison)

    - 1 !CLW (for identification and blindness)
    - 1 + !CSW (for identification and confusion)

a. Buffing

    - 1 !Hero (for protection from fear.)

a. Ranged attack to soften up a (single) unique

    - ~5 Flasks of Oil to kill dangerous and/or valuable monsters (throw oil for damage with the `v` throw command)
    - ~10-20 Iron shots (or arrows for Rangers) to throw at non-dangerous monsters with annoying side effects. (Stat drainers and acid damagers that are between you and the stairs down.) extra arrows for a ranger (shoot arrows with the `f` fire command.) Shots can be reused; oil can't.

a. Armor

Don't bother buying armor - it's very expensive in comparison to AC. You will find it in the dungeon soon enough.

Your starting equipment will include more than enough food and illumination for the first trip down.

The first trip

It's quite possible to get to 500' (dl 10) or deeper in the first trip into the dungeon. Plan to return when you run out of either escapes (?Phase door), protection from side-effects (curing potions), or damage (arrows if you are a ranger, flasks of oil if you are otherwise weak.)

Since you don't have much, don't spend it on less valuable monsters.

What to kill

- Any mobs of monsters that you can defeat (kill or frighten) in a single turn and either give good experience (a pack of wolves) or good drops (weaker orcs, novice humans.) If you are deep enough, this is likely to increase your character by several levels in a single battle.
- Uniques with good drops (Bullroarer, Brodda, Wormtongue)
- Easy kills that are likely to drop something worthwhile.

What to ignore

- Monsters that will damage or destroy gear. (jellies, water hounds, etc)
- Non-valuable monsters that are likely to use up consumables (baby gold dragons, groups of spell-casters in line-of-sight, etc)
- Mobs that you can't dominate.
- Uniques with escorts you can't dominate
- Monsters at shallow depth. Drops for any given monster get better the deeper you go. Killing a novice mage at dl 1 generally gives nothing; at dl 20 he's likely to drop something worth hundreds of gold. Wormtongue has, on average, a noticeably better drop at dl 20 than dl 10.
- Things that waste effort. (Run-away breeders, low-EXP monsters with no drop. Just close the door and move on.)

What to avoid

- Anything that can kill you in a single turn.

On Bad Luck

This is rule number one of Angband: don't take unnecessary risks. If you take enough low-probability chances of death, you'll never survive to fight Sauron. Such deaths are generally called 'stupid', but that's not always accurate. Sometimes it's just bad luck. But given enough chances, you are guaranteed to receive it. It's the trick to extremely fast dives: the fewer moves you make, the less chance

any one of them will be fatal, even if on average, your individual moves are riskier than in slower play. But the strategy applies more generally: unless you are exceedingly careful in play, messing around long enough at any one depth guarantees that something bad will eventually happen.

Face-palm Tips

The preceding is good advice; however, it does not offer much more than generalizations, albeit valid, for the “intermediate” (?) player. The following is intended to state what many perceive to be blatantly obvious, hence “you did WHAT!” face-palm deaths. This really should be cleaned up and refactored, but placing here for now.

WARNING. I have yet to defeat Angband. This is a compilation of some of the better tips I’ve learned while trying to explore the depths… (to Level ~35). Additional advice would be greatly appreciated!

Start simple

Begin your Angband career as a warrior. Warriors are relatively simple to begin with, and are less likely to be eaten by a pack of jackals.

Focus!

Angband is a very harsh game, in that the character you’ve been playing for months could be killed by a single careless action. Playing when tired or drunk is probably a good way to leave yourself with nothing but a sad tale to share on the forums. If you have the ability to sense monsters or traps then use it. This is particularly important when you begin to encounter monsters that, without the appropriate gear, will kill your character in a single move.

Use that stuff

Angband has potions, spell books, wands, staffs, rods, activate-able items, melee weapons, ranged weapons, and whatnot. They're meant to be used, for crying out loud! It can take a while to get used to using all the different types of items, but they work best when used in concert. For those able to use magic devices, rods wands and staves can be very useful when your mana is running low, and allow you access to spells that may not normally be available to your class. They are also very useful for dealing with monsters that

Rangers have a bow

Really a subset of the previous point, but it happens so often… Don't try to play a ranger like a warrior – rely on the bow! Similarly for mages, don't do a Gandalf. He may be able to draw a sword and rush headlong in to a pack or orcs, but mages in Angband are considerably more fragile. Priests are better equipped to engage in melee combat - with their healing abilities compensating for their somewhat fragile nature.

Stockpile!

Players may not be inclined to carry multiples of an item, or do so in a limited quantity, perhaps due to weight encumbrance concerns. Don't be afraid to carry a LOT of an item, particularly the basics – food, light, projectiles, cure potions, "run away" scrolls, etc. Don't be afraid to MAX OUT important items! Some monsters will steal or destroy your items, so it's worth carrying additional quantities of key items - such as Scrolls of Recall or important spell/prayer books. This becomes very important when you reach dungeon levels in which monsters develop fire and acid-based attacks. Mages and priests tend to start out with low strength, and so are very limited in how much stuff they can carry, so consider carrying additional copies of the spellbooks that you know you can't afford to lose during a fight.

An item you don't use is useless

Common fallacy: you find an incredibly powerful Staff of Mighty BOOM! (3 charges), or a single Potion of become Chuck Norris, and

then you keep carrying it around and never actually use it. It could be a life insurance, but you might still reconsider your strategy: maybe you've become too careful lately (Angband rewards deliberate risk-taking, after all).

Identifying your items

Many items found, especially early in the game, will be unidentified or partially identified. Weapons and armour can be identified by being worn and used in combat, and can be removed if found to be of poor quality. Some characters learn to cast spells to identify the runes on wearable items that define their properties.

Staves, rods and wands can often be identified by being used against monsters, but be aware that some of these magical items can have negative effects. Only use this approach when facing easily defeated monsters. Consuming unidentified potions and mushrooms can be risky, so the risk-averse player may prefer to sell them in the town. Ammunition can normally be identified by being thrown or fired at a monster, and typically the worst thing that can happen is that the attack does little damage to the monster. As with staves, wands and rods, do this when facing a monster than can be easily defeated.

Scrolls can normally be identified by being read, but some scrolls have negative effects. Your character may have a very short life if they read a scroll that summons a horde of undead monsters. If using this approach, it's a good idea to position your character on top of some stairs so you can quickly escape if a mysterious scroll leaves your character surrounded by monsters.

With weapons and armour, your character will in time learn their inscribed runes.

The dungeon is dark

Players will readily note that corridors are unlit; however, what may not be blatantly obvious is that the dungeon gets darker with depth, until it's pitch black. ALWAYS carry a light source. DON'T drop a light source for more loot!

You can RUN AWAY

There is NO RULE that you have to clean out a dungeon – AT ALL. See something you can't handle? Don't be afraid to leave – NOW. Sure you could try to avoid it, but then again, it could be a hummerhorn. Half way through the level? You can still LEAVE NOW. Nearly finished with the level? You can still leave NOW. Think of it as a tactical strike, not a genocide mission.

Get away NOW!

Sure, you should pay more attention and not get yourself in a next turn = death situation, but this is not always avoidable and no one (presumably) is so meticulously patient. ALWAYS carry get away items – Phase Door, Teleport, Recall, Teleport Level, etc. When you can, stockpile those that work NOW. If a unique starts chasing you, you don't want to be waiting for Recall to kick in.

You should seriously consider a quick escape if dealing with a situation in which monsters are breeding explosively. Some classes, such as mages, are more easily able to handle these though the use of spells that affect multiple monsters simultaneously, but even they should consider leaving if it becomes apparent that the monsters are breeding more quickly than the player can kill them.

Level 1 is still there

Don't forget that you can always replay, and re-re-play the early levels for ANY reason whatsoever. Recalled from a depth too deep? Dive from level 1 and reset the depth. Want to fill your armor slots? Need a few more gold? Even if you're on a streak diving through the dungeon, step back and reassess whether you want to return to an earlier level – you never know what could be lurking around the corner.

Black Market Deals

Don't be afraid to buy from the "blackmarket". Sure the prices are more expensive than the other stores, but it tends to offer a good selection of items and it can be worth the gold versus not having it

in the dungeon. You can always get more gold, but the RNG is random. It's just another store – don't worry about the name. The blackmarket is often a good source of potions to increase your stats, so it's worth checking it each time you visit the town - and try to have enough gold to purchase potions for the most important stats for your character.

Try a Different Strategy

Sure, what you're doing kinda works, but your characters keep dying off. Playing conservatively? Try playing a character as though the "iron-man" stairs setting is in effect. You'll probably want some ranged weapons for this, but you may be surprised how fast you can level if you DIVE! Granted, the lower levels are harder, but the deeper you go, the more experience per kill you get. Still fight conservatively, but dive aggressively. Once your character has gotten a decent complement of items and spells … (what next?). Don't be afraid to throw away characters – DIVE aggressively until you get a good feel for your character and you've leveled up.

On Depth and Item Value

Whenever you kill a monster that may drop an item, a lot of randomness is played out. No need to go into detail just here (if you really want to know, you can look it up in the spoilers), suffice it to say that depth is the single most important factor. You may, of course, find rather valuable and/or useful stuff even on the very first level, but the chances are rather small.

In short: if you want to find better things, you need to go deeper.

The monster barely matters

If a monster drops an item, it can be anything. Really. You may find rare spellbooks on illiterate Trolls, and jellies may drop artifact weapons. Some rare monsters, like powerful dragons, will carry good or even exceptional objects – but still, the dungeon level is the most important factor. What was "exceptional" on dl8 will probably be rather uninteresting on level 30.

Regarding Artifacts

For every single wearable item, there is a small chance that it will be an artifact. This might seem slim, but considering the number of monsters you will slay, it soon adds up. If you work through a large room full of orcs, there's about a one-in-ten chance that you'll discover at least one artifact among the carnage. Artifacts typically have impressive statistics and an ability that can be activated.

Surviving to clvl 31 / dlvl 36

Additional tips learned from various failed attempts. Still have not beat angband, but making considerable progress of late, particularly with my (now) current character.

RUN AWAY!

Very first tip is a basic tenet repeated throughout the guide; however, it is far too easy to become engrossed in your achievements (eg an awesome artifact, maximizing stats, a slightly better item, gaining a second blow, getting a speed bonus, etc) and lose it all due to something very avoidable had you been paying diligent attention.

While playing, always make sure to check for the basics – particularly monsters, stairs, and traps. Don't lose a character, asserting that you can kill a "C" easily only to discover that a "wild dog" is a far cry from a "hellhound".

Similarly, if you just got some nice artifacts / other equipment or achieved a significant accomplishment – save your game and take a break. You don't have to continue to play to enjoy your past success, and when you return you will likely be more able to play diligently.

See a few uniques or other monsters that might possibly give you some trouble – maybe? Perhaps with only slight difficulty? See a vault with some cool treasure, but a few troublesome monsters / uniques? If you're not highly confident (perhaps even 100%) that you can easily survive the level, you can leave on the spot. Just

enter the level, immediately do your detection routine (monsters, stairs / traps, etc) assess the immediate threat, and use the stairs immediately if you want to. You can go up as readily as you came down, and you can go down as readily as you came up. Further, artifacts can be regenerated (unless you disabled that option) and as you dive, even better items will be available. So, if you skip out on levels just because you want to play it very conservatively, you will still find great items and better yet, you have a higher chance of survival.

Information Awareness

Angband offers a LOT of information on a LOT of different screens. If you haven't already, try enabling multiple consoles – ALL of them – and then try out different options / combinations. Being able to view a lot of information simultaneously at a glance is a considerable benefit over having to be diligent enough to manually check each relevant screen each time.

Also, change the text size. Sure you may be accustomed to your terminal font and size for reading; however, you can read and make sense from context a lot easier than you can reliably identify the glyph, color, and relative position of every character on the screen. Angband may be a text-based game, but you don't have to play with a tiny (or even normal size font). Using a slightly larger font makes identifying things a LOT easier. The SDL interface is perhaps the easiest for using multiple terminals and changing fonts.

This alone has greatly contributed to my survivability!

Start from the Beginning

Trying to mature your character through the clvl / dlvl 20s can be challenging until you can survive even lower depths. One tip here is to use Recall exclusively to return from the dungeon and use the stairs to return back down to the level you left.

Yes, this is more work, and as your clvl increases, the early levels become exceptionally easy; however, it guarantees that you will always enter a level on stairs, facilitating running away.

Item Collecting

Its easy to stockpile, but you'll run out of space VERY quickly. Further, what was useful may not be useful now and what's useful now may not be useful later. It can be easy to try to grab everything in sight and then return to town when your inventory is full; however, when you reach this point, try continuing to dive, even with a full inventory.

As your character progresses and you continue to dive you will find better classes of items. Don't be afraid to toss the less valuable stuff to make room, so you can continue diving.

This is particularly important while diving. If you let a full inventory be a limiting factor, you won't be able to get nearly as deep as if you are willing to discard items along the way.

Take a Break

Again, this is another basic tip repeated thorough out the guide; however, learn to stop playing. You can always save your game and resume later when you would be in your prime and optimally ready to play as opposed to trying to continue to play as drowsiness sets in.

This may be difficult at first, but it can help you avoid running decent characters into the ground for avoidable mistakes…

Don't forget your Ranged Attacks!

Granted, as your character develops, you become more powerful at melee; however, this does not preclude you from using ranged attacks! It is far to easy to start playing a ranger like a warrior mage if you get a good melee weapon, but don't forget your ranged skill! You may be "Superb" at Fighting, but ranger's are even better at Shooting – perhaps even "Legendary" or better.

This can be a hard one to do diligently, especially if your melee weapon is powerful, but try to play diligently and use ranged attacks. Try earlier levels where survivability is much higher and

try to play without melee attacks for awhile to get used to using the bow again.

Even as your character matures, try to keep in mind your race / class core strengths.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Frequently Asked Questions

The best way to get answers to your questions is to post them on the Angband forum [http://angband.oook.cz/forum].

Contents

Issues and problems

How do I report a bug?

Post on the Angband forum [http://angband.oook.cz/forum].

Bug reports should include:

- your current operating system (e.g. Windows 10)
- what version the problem appeared in
- the best steps you can figure out to reproduce the bug.

Savefiles that show the problem might be requested, because they help tracking bugs down.

Dark monsters are hard to see

Fix (reduce) the alpha on your screen, or use the “Interact with colors” screen under the options (=) menu. Navigate to the `8` using `n` and increase the color intensity with r(ed)/g(reen)/b(lue).

Is there a way to disable that thing that pops up when you hit the enter key?

Go into the options menu, choose “Edit keymaps”, then “Create a keymap”. Press Enter at the “Key” prompt and a single space as the “Action”.

And then you’ll probably want to choose to “Append keymaps to a file” so that it persists for next time you load the game.

This just replaces the default action of Enter with a “do nothing but don’t tell me about help” action. If you want to keep the menu available, say on the ‘Tab’ key, you can also remap the Tab keypress to the `\n` action.

Development

What are the current plans for the game?

Ever-changing and subject to U-turning with public dissent. Discussion typically happens in the Vanilla section [http://angband.oook.cz/forum/forumdisplay.php?f=3] of the forums.

How do I suggest an idea/feature?

Post it on the forums. If people think it’s a good idea, it will generally get some discussion; if they don’t, it won’t. The developers keep an eye on the forums, and ideas deemed OK will get filed for future implementation.

Sometimes a suggestion may not be right for the game, though. Some suggestions would change aspects of Angband that are essential to its nature; Angband has a long history, and so has developed a certain character over the years. Some suggestions might make a good game, perhaps even a better game than Angband, but would make a game that is not Angband. To some extent, variants exist to address this, but even so they tend to adhere to the core Angband principles.

How do I get a copy of the source code?

Go to the GitHub [https://github.com/angband/angband/] page, where you can find the bleeding edge as well as all previous versions.

How do I compile the game?

Please see the compiling section of the manual.

How do I contribute to the game?

You have two options:

- Write your patch and submit it as a pull request on GitHub.
- Post about it on the forum.

All contributions are accepted as dual-licenced with both the Angband and GPLv2 licences.

There are contribution guidelines in CONTRIBUTING.md in the top level directory of the source code.

If the patch is a bugfix, then you can reasonably expect it to be integrated into the development tree. If it's more involved, and the feature is not one that the next version is planned to have, the patch may go through several reviews before being incorporated. It may also just be unsuitable for Angband - in which case, please don't take rejection badly; you may just be better off writing a variant.

Non-code activities are different. Documentation can be written on the wiki, or if you're a graphics designer (and they're always

welcome) then please talk on the mailing list about your work.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Creating a Character

Angband is a roleplaying game, in which you, the player, control a character in the world of Angband. Perhaps the most important thing you control is the birth of your character, in which you choose or allow to be chosen various attributes that will affect the future life of your character.

At the character creation screen you will be prompted to select the race and class of your character. You also have the option to change the 'birth options' at this time. These need to be set at the character creation menu and cannot be altered later in the game. They are discussed with the rest of the options in the "options" help file.

Character Characteristics

Each character has two primary attributes: race and class. These are chosen at the beginning and which will stay fixed for the entire life of that character. The race and class have many effects which are discussed in detail below.

Each character has a few secondary attributes: height, weight, and background history. These are randomly determined according to the race of the character, and are only used to provide flavour to the character to assist in role playing. There is an opportunity to edit background history during character birth.

Each character also has five primary "stats": strength, intelligence, wisdom, dexterity, and constitution; they modify the abilities of the character in a variety of ways. Every stat has a numerical value, ranging from a minimum of 3, up to a normal maximum of 18, and

even higher, into the “percentile” range, represented as ‘18/01’ through ‘18/100’: this is the maximum that can be achieved intrinsically, for any given stat. These stats can be modified further by equipment, race and class bonuses up to a maximum of ‘18/220’.

Each character also has several primary “skills”: disarming, magic devices, saving throws, stealth, searching ability, fighting skill, shooting skill, and digging skill, which are derived from the character’s race, class, level, stats, and current equipment. These skills have rather obvious effects, but will be described more completely below.

Each character may have one or more “racially intrinsic skills”, based on the race of the character. These may include special resistances, or abilities such as infravision.

Each character has a number of “experience points”, which increases as the character defeats monsters and attempts new spells and uses new items. Characters also have a level, which is based on experience. The amount of experience required to gain a new level is dependent on the character race and class. Races and classes with more intrinsic benefits require more experience to gain levels. As the experience rises, so does the level, and as the level rises, certain other abilities and characteristics rise as well. All characters start at 0 experience and at the first level.

Each character has some gold, which can be used to buy items from the shops in the town, and which can be obtained not only from selling items to the shops, but also by taking it from dead monsters and by finding it in the dungeon. Each character starts out with some gold, the amount of which is based on the character’s social class (higher is better) and stats (less powerful characters start with more gold). Each character also starts out with a few useful items, which may be kept, or sold to a shop-keeper for more gold. However, especially valuable items will never sell for the full price, as each shopkeeper has a maximum that they are prepared to pay for any item. The more generous shopkeepers may buy your items for up to 30,000 gold pieces: but some are really stingy, and will pay no more than 5,000.

Each character has an “armour class”, abbreviated to AC,

representing how well the character can avoid damage. The armour class is affected by dexterity and equipment, so the concept includes both avoiding blows and being able to take blows without being hurt. Armour class on equipment is always denoted in square brackets, usually as a figure of '[X, + Y]' where X is the intrinsic AC of the armour in question, and Y is the magical bonus to armour class provided by that item.

Each character has "hit points", or hp, representing how much damage the character can sustain before they die. How many hit points a character has is determined by race, class, level and constitution, as follows: each race has a basic "hit dice" number - for instance, a Dwarf's basic hit die is 11, while a hobbit's is 7. This is modified by class: for instance, a warrior gets a $+9$ bonus to the hit die, while a mage gets no bonus and a priest $+2$, so a dwarven warrior's total hit die will be $(11+9)=20$ - meaning that they get between 1 and 20 hit points per level. If they were a priest, his hit dice would be $(11+2)=13$, and get between 1 and 13 hit points per level. The hobbit mage would get only 1-7 hps per level. (All characters get the maximum at first level: thereafter it is a random roll of 1dX where X is the hit die number, when the character goes up in level.) This is further modified by constitution - a character with high constitution will get a flat bonus of a certain number of hit points per level (recalculated right back to level 1: if you gain an extra hit point per level, and you are 42nd level, you will suddenly be 42 hit points better off.)

Each character has spell points, or mana, which limits how many spells (or prayers) a character can cast (or pray). The maximum number of spell points is derived from your class, level, and intelligence (for spells) or wisdom (for prayers), and you can never have more spell points than the maximum. Spell points may be regained by resting, or by magical means. Warriors never have any spell points. If a character gains enough wisdom or intelligence to get more spell points, the result is calculated right back to first level, just as with constitution and hit points.

Lastly, each character has a base speed. Speed determines the amount of "energy" your character acquires in the game, and therefore how often you can take actions which use up energy (like

moving or attacking). All beginning characters move at normal speed and the only way to increase speed is by magical means and equipment bonuses. Characters who are carrying too much weight will move more slowly. Extra speed is one of the most important boons in the game and therefore one of the rarest and most sought after.

Races

There are eleven different races that you can choose to play in Angband. Each race has its own adjustments to a character's stats and abilities. Most races also have intrinsic abilities. The bonuses to statistics and the experience penalty will be displayed next to the races as you move to select one.

Human

> The human is the base character. All other races are compared to them. Humans are average at everything and tend to go up levels faster than any other race due to their shorter life spans. No racial adjustments or intrinsics occur to human characters. Humans do not have any infravision.

Half-Elf

> Half-elves tend to be smarter and faster than a human, but not as wise or strong. Half-elves are slightly better at magic, disarming, saving throws, stealth, archery and searching, but they are not as good at hand-to-hand combat. Half-elves are immune to dexterity draining, and have weak infravision.

Elf

> Elves are better magicians than humans, but not as good at fighting. They tend to be smarter and faster than humans, though not as wise or strong. Elves are better at searching, disarming, perception, stealth, archery and magic, but they are not as good at hand-to-hand combat. They are resistant to attacks involving bright light, are immune to dexterity draining, and have fair infravision.

Hobbit

Hobbits, or Halflings, are very good at shooting, throwing, and have good saving throws. They also are very good at searching, disarming, perception, and stealth; so they make excellent rogues, but prefer to be called burglars. They will be much weaker than humans, and not good at melee fighting. Halflings have fair infravision. They have a strong hold on their life force, and are thus resistant to life draining. Hobbits are very partial to mushrooms and can identify them when found.

Gnome

Gnomes are smaller than dwarves but larger than halflings. They, like the halflings, live in the earth in burrow-like homes. Gnomes make excellent mages, and have very good saving throws. They are good at searching, disarming, perception, and stealth. They have lower strength than humans so they are not very good at fighting with hand weapons. Gnomes have good infravision. Gnomes are intrinsically protected against paralysis and some slowing effects. Gnomes are excellent at using wands and staves and can identify them when found.

Dwarf

Dwarves are the headstrong miners and fighters of legend. Since dungeons are the natural home of a dwarf, they are excellent choices for a warrior or priest - or indeed, that combination of the two, the paladin. Dwarves tend to be stronger and tougher but slower and less intelligent than humans. Because they are so headstrong and are somewhat wise, they resist spells which are cast on them. Dwarves also have excellent infravision. They can never be blinded. Dwarves are excellent at digging, and can sense nearby buried treasure. They have one big drawback, though. Dwarves are loudmouthed and proud, singing in loud voices, arguing with themselves for no good reason, screaming out challenges at imagined foes. In other words, dwarves have a miserable stealth.

Half-Orc

Half-Orcs make excellent warriors and decent priests, but are

terrible at magic. They are as bad as dwarves at stealth, and horrible at searching, disarming, and perception. Half-Orcs are, let's face it, ugly. They tend to pay more for goods in town. Half-Orcs do make good warriors and rogues, for the simple reason that Half-Orcs tend to have great constitution and lots of hit points. Because of their preference to living underground to on the surface, half-orcs resist darkness attacks. They have fair infravision.

Half-Troll

Half-Trolls are incredibly strong, and have more hit points than any other character race. They are also very stupid and slow. They will make great warriors and iffy priests. They are bad at searching, disarming, perception, and stealth. They are so ugly that a Half-Orc grimaces in their presence. Half-trolls always have their strength sustained, and they regenerate quickly like other trolls. Unfortunately, this regeneration also requires them to eat more food than other races. They have fair infravision.

Dunadan

Dunedain are a race of hardy men from the West. This elder race surpasses human abilities in every field, especially constitution. Their hardiness ensures that their constitution cannot be reduced. They have no infravision.

High-Elf

High-Elves are descended from those among the Elves who heard and answered the call from the Valar at the very beginning of time, before the sun and moon were made, and lived in the Blessed Realm for many thousands of years before returning to mortal lands. Because of this, they are far superior in terms of abilities when compared to their lesser Elven kindred. They can also see into the invisible world of ghosts and wraiths. However, they find new experience harder to come by than other races. Like normal Elves, they resist attacks involving bright light. They have good infravision and can even see cold-blooded invisible creatures.

Kobold

Kobolds are a race of small dog-headed humanoids that dwell underground. They have excellent infravision, and are intrinsically resistant to poisons of all kinds. They have a good dexterity and constitution. However, they are weaker than humans, and also not noted for great intelligence. Furthermore, they are ugly, and not trusted in town. And while their constitution makes them tough, it still cannot prevent the fact that they are not the biggest of creatures, and have few hit points.

Classes

Once a race has been chosen, you will need to pick a class. The class is the character's occupation and determines stat bonuses, abilities, hit dice, and what spells (if any) the character can learn.

Warrior

A Warrior is a hack-and-slash character, who solves most problems by cutting them to pieces, but will occasionally fall back on the help of a magical device. Their prime stats are strength, constitution, and dexterity, and they will strike more blows with melee weapons than any other class. A Warrior will be excellent at fighting, shooting and throwing, but bad at most other skills. A warrior has bad stealth and cannot learn any spells.

Mage

A Mage must live by their wits. They cannot hope to simply hack their way through the dungeon, and so must therefore use his magic to defeat, deceive, confuse, and escape. A mage is not really complete without an assortment of magical devices to use in addition to his spells. They can master the higher level magical devices far easier than anyone else, and has the excellent saving throws to resist effects of spells cast at him. However, they are incredibly weak, getting few hit dice and suffering strong penalties to strength and constitution. Intelligence is their primary stat and at high levels they can cast many spells without a chance of failure.

There is no rule that says a mage cannot become a good fighter, but spells are their true realm and they will get fewer blows with melee weapons than other classes.

Druid

A Druid is a lover of nature, and at one with the natural world. Druids have control of their environment and direct power over creatures, leading even to the ability to take their forms. Druid skills are moderate, and they have some fighting ability, especially when transformed. A druid will usually seek to shape the flow of events to their purpose rather than using devices or missiles, but at high levels they do gain access to potent nature-based magic. Druids rely on their wisdom, and have good saving throws.

Priest

A Priest is a character of holy devotion. They explore the dungeon only to destroy the evil that lurks within, and if treasure just happens to fall into their packs, well, so much more to the glory of their temple! Priests receive their spells from a deity, and therefore do not choose which spells they will learn. They are familiar with magical devices, preferring to call them “instruments of God”, but are not as good as a mage in their use. Priests have great saving throws, and make passable fighters, better if they can find a blessed weapon. Wisdom is the priest’s primary stat and at high enough levels they can cast many prayers without a chance of failure. Priests have poor stealth.

Necromancer

A Necromancer seeks to master the spirits of sentient creatures, creating servants of their own will. They have chosen a dark and wicked path, and run a continual risk of harm to their own body and mind. Necromancers love shadows and hate light, automatically shrouding themselves in darkness. Their spells require high intelligence, and frequently harm the caster or place them in danger. In return they achieve awesome power at high levels. The ultimate aim of the necromancer is to supplant the Dark Enemy and set up a tyranny in his place.

Paladin

A Paladin is a warrior/priest. Paladins are very good fighters, second only to the warrior class, but not very good at missile weapons. They receive prayers at a slower pace than the priest, but can use all but the most powerful prayers. A paladin lacks much in the way of abilities. Paladins have poor stealth, perception, searching, and magical device use. They have a decent saving throw due to their divine alliance. Their primary stats are strength and wisdom.

Rogue

A Rogue is a character that prefers to live by their cunning, but is capable of fighting their way out of a tight spot. The master of traps and locks, to the experienced rogue no device is impossible to overcome. A rogue has a high stealth enabling sneaking around many creatures without having to fight, or sneaking up and get the first blow. They can steal items from monsters, but need to be wary of being caught in the act. Rogues' perception is higher than any other class, and many times they will notice a trap or secret door before having to search. A rogue is better than the more fighting oriented classes with magical devices, but still cannot rely on their performance. Rogues can also learn a few spells, but not the powerful offensive spells mages can use, and there will always be a chance of failure even with the simplest spells. A rogue's primary stats are dexterity and intelligence.

Ranger

A Ranger is at home in nature, and good at turning the environment to good use. Rangers are good fighters, and the best of all the classes with missile weapons, especially bows. The ranger learns chiefly spells of survival and forest craft. A ranger has good stealth, good perception, good searching, a good saving throw, and is good with magical devices. Their primary stats are strength, wisdom and dexterity.

Blackguard

A Blackguard is a brawler of no principle, who lives for the joy of maiming and killing. Blackguards prefer heavy weapons and shields, and learn a few spells for the purpose of

wreaking more destruction. Their lust for blood is legendary, with a blackguard in full cry nigh on impossible to kill. Blackguards scorn skills apart from slaughter, and require strength and intelligence (or rat-cunning) to thrive in the dungeon.

Stats

After race and class are selected, you will be able to decide what stat levels your character will have, by allocating a finite number of "points" between the five statistics. These points can be allocated by selection or with a random roller (as described below in the "Stat Rollers" section). Each race/class combination also has a recommended default setting for these statistics. Statistics can be permanently raised by various potions in the dungeon up to 18/100. They can also be temporarily drained by some monster attacks.

Strength

Strength is important in fighting with weapons and in melee combat. A high strength can improve your chances of hitting as well as the amount of damage done with each hit. Characters with low strengths may receive penalties. Strength raises the amount of weight you can carry before being slowed. It also allows you to get extra blows with heavier weapons. Strength is one of the most important stats in the beginning of the game.

Intelligence

Intelligence affects the spellcasting abilities of spellcasters from the arcane and shadow realms: mages, rogues, necromancers and blackguards. Intelligence will affect the number of spells you may learn each level as well as the number of spell points you receive. Intelligence is the most important stat for mages and necromancers. A high intelligence may also improve your chances of successfully casting a spell. You cannot learn spells if your intelligence is 7 or lower. A good intelligence can also help with using magic devices, picking locks, and disarming traps.

Wisdom
: The primary function of wisdom is to determine the ability of a priest or paladin to use prayers, and druids and rangers to use verses, just like intelligence affects spellcasting. Again, high wisdom will increase the number of mana points you have and increase the number of prayers or verses you can learn each level, while improving your chance of success. A good wisdom increases your saving throw, thereby improving your chances of resisting magical spells cast upon you by monsters.

Dexterity
: Dexterity is a combination of agility and quickness. A high dexterity may allow a character to get multiple blows with lighter weapons. Dexterity also increases a character's chances of hitting with any weapon and of dodging blows from enemies. Dexterity is also useful in picking locks, disarming traps, protecting yourself from some of the thieves that inhabit the dungeons, and (for rogues) stealing successfully from others. If the character has a high enough dexterity, thieves will never be successful in stealing from them.

Constitution
: Constitution is a character's ability to resist damage to his body, and to recover from damage received. Therefore a character with a high constitution will receive more hit points and also recover them faster while resting. Constitution is less important in the beginning of the game, but will be the most important stat at the end.

Skills

Characters possess some different skills which can help them to survive. The starting skill levels of a character are based upon race and class. Skill levels may be adjusted by high or low stats, and may increase with the level of the character.

Infravision
: Infravision is the ability to see heat sources. Since most of the

dungeon is cool or cold, infravision will not allow the player to see walls and objects. Infravision will allow a character to see any warm-blooded creatures up to a certain distance. This ability works equally well with or without a light source. However, some of Angband's creatures are cold-blooded, and will not be detected unless lit up by a light source. All non-human races have innate infravision. Humans (including Dunedain) cannot gain infravision unless it is magically enhanced. Infravision does not increase with character level, and is purely dependent on race and on magical equipment.

Fighting

Fighting is the ability to hit and do damage with weapons or fists. Normally a character gets a single blow from any weapon, but with high enough dexterity and strength may receive more blows with lighter weapons. Strength and dexterity both modify the ability to hit an opponent. This skill increases with the level of the character. Inspecting a weapon will show you how quickly you can attack with it.

Shooting Ability

Using ranged missile weapons (and throwing objects) is included in this skill. Different stats apply to different weapons, but this ability may modify the distance an object is thrown/fired, the amount of damage done, and the ability to hit a creature. This skill increases with the level of the character.

Saving Throws

A Saving Throw is the ability of a character to resist the effects of a spell cast on him by another person/creature. This does not include spells cast on the player by his own stupidity, such as quaffing a nasty potion. This ability increases with the level of the character, but then most high level creatures are better at casting spells, so it tends to even out. A high wisdom also increases this ability. It is possible to get 100% saving throw, making you immune to many attacks.

Stealth

The ability to move silently about is very useful. Characters with good stealth can usually surprise their opponents,

gaining the first blow. Also, creatures may fail to notice a stealthy character entirely, allowing a player to avoid certain fights. This skill is based upon race and class, but can be magically enhanced.

Disarming

Disarming is the ability to remove traps safely, and also includes picking locks on traps and doors. A successful disarming will gain the character a small amount of experience. A trap must be found before it can be disarmed. Traps are either physical or magical devices, so the character has two disarming skills. Dexterity modifies disarming of physical traps, and intelligence modifies disarming of magical traps. Both these abilities increase with the level of the character.

Magical Devices

Using a magical device such as a wand or staff requires experience and knowledge. Spell users such as mages and priests are therefore much better at using a magical device than say a warrior. This skill is modified by intelligence, and increases with the level of the character.

Searching (Perception)

Perception is the ability to notice traps without actively seeking them out. Rogues are the best at searching, but rangers are also good at it. This skill is based on race and class, and increases with character level.

Stat Bonus Tables

Stats, hit dice, infravision and experience point modifications due to race and class are listed in the following table. To get the total hit dice, add the “race” and “class” numbers: for instance, a Dwarf Priest has a hit die of $11 + 2 = 13$ (i.e. they will get 1d13 hit points per level, adjusted for constitution).

[illegible]
Human [illegible] 100%
Half-Elf [illegible] 110%

[illegible]	[illegible]%
[illegible]	[illegible]it
[illegible]	[illegible]me
[illegible]	[illegible]f
[illegible]	[illegible]Orc
[illegible]	[illegible]Troll
[illegible]	[illegible]adan
[illegible]	[illegible]h-Elf
[illegible]	[illegible]old
[illegible]	**[illegible](bonus)**
[illegible]	[illegible]arrior
[illegible]	[illegible]age
[illegible]	[illegible]id
[illegible]	[illegible]est
[illegible]	[illegible]romancer
[illegible]	[illegible]adin
[illegible]	[illegible]gue
[illegible]	[illegible]nger
[illegible]	[illegible]ackguard

Ability Tables

[illegible]	[illegible]
[illegible]	[illegible]uman
[illegible]	[illegible]alf-Elf
[illegible]	[illegible]lf
[illegible]	[illegible]obbit
[illegible]	[illegible]nome
[illegible]	[illegible]warf
[illegible]	[illegible]alf-Orc
[illegible]	[illegible]lf-Troll
[illegible]	[illegible]unadan
[illegible]	[illegible]gh-Elf
[illegible]	[illegible]obold

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]	[illegible]ncer
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]rd

For character classes, there are two figures: the first figure is the base level of the skill, while the second figure (in parentheses) is the bonus that the character gains to this skill every ten levels. So, to find out the total skill value of any character's skills, add the race value to the class value, and then the bonus once for every ten levels that the character has.

Please note, however, that these numbers are only good for comparing characters to each other in the absence of other bonuses from high stats (strength bonus to-dam, dex bonus to-hit, wisdom bonus to saving throw, intelligence bonus to magical device usage, etc.) or wearing magical items.

Stat rollers

There are currently two different ways to determine the starting stats of your character - you can choose which one to use from the birth screen.

Point-based

> The point-based method allows you to "buy" improvements to your basic stats by "spending" points on them. You have a fixed number of points to spend, and making small changes to a stat costs proportionally less than making large changes. Any unspent points are converted into your starting money that you can use to buy equipment at the start of the game.
>
> On selecting this option, you will find that the points have already been assigned to default recommended values. These represent an algorithm's opinion for the ideal point spending. However, you are free to reallocate them as you wish.
>
> This is the recommended birth method.

Standard roller

The standard roller simply rolls three six-sided dice for each stat, leaving everything to chance. You can press `r` to re-roll the dice, or simply accept what luck has offered.

Character Name

Once you have accepted a character you will asked to provide a name for the character. In general, the actual choice of a name is not important, but do keep in mind that it may have some effect on the game itself. For example, on some machines, the character name determines the filename that will be used to save the character to disk. The character name is used on the high score list.

You can play a dynasty of characters. If you use a Roman numeral at the end of your character name (like “Fred I” or “Pimplesnarg XVI”), the game will automatically increment the numeral each time you die (or win!).

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Exploring the Dungeon

After you have created your character, you will begin your Angband adventure. Symbols appearing on your screen will represent the dungeon's walls, floor, objects, features, and creatures lurking about. In order to direct your character through their adventure, you will enter single character commands (see 'commands.txt').

Symbols On Your Map

Symbols on your map can be broken down into three categories: Features of the dungeon such as walls, floor, doors, and traps; Objects which can be picked up such as treasure, weapons, magical devices, etc; and creatures which may or may not move about the dungeon, but are mostly harmful to your character's well being.

Some symbols are used to represent more than one type of entity, and some symbols are used to represent entities in more than one category. The "@" symbol (by default) is used to represent the character.

It will not be necessary to remember all of the symbols and their meanings. The "slash" command (/) will identify any character appearing on your map (see 'commands.txt'), and there is a comprehensive menu of terrain, objects, monsters and so on using the "tilde" (~) command.

Note that you can use a "user pref file" to change any of these symbols to something you are more comfortable with.

Features that do not block line of sight

.	An open space
<	Entrance to General Store
(	Entrance to Armoury
)	Entrance to Weapon Smith
^	A glyph of warding
^	Entrance to Bookseller
'	An open door
'	Entrance to Alchemy Shop
+	A broken door
+	Entrance to Magic Shop
<	Staircase up
<	Entrance to the Black Market
>	Staircase down
>	Entrance to your Home
≈	A pool of lava

Features that block line of sight

#	A wall
#	A secret door
+	A closed door
+	A locked door
*	A locked door with treasure
*	A pile of passable rubble

Objects

/	A polearm (or flask)
\	A sceptre (or wand)
|	A sword (or rapier)
!	A potion (or bottle)
?	A scroll (or paper)
,	A mushroom (or food)
~	A staff
{	A sling, bow, or x-bow
(	A shot, arrow, or bolt
[	Soft armour
[	Hard armour
[	Boots, gloves, or cloak
[	A shield
]	Hats, Helms, or crowns
=	Rings, Amulets, or Lights
"	Amulets or necklaces
-	Wands or rods
_	Staves
$	Gold or gems
/	Digging Tools, Chests, etc
&	Multiple items

Monsters

a	Giant Ant
b	Giant Bat
c	Giant Centipede
d	Dragon
e	Floating Eye
f	Giant Frog
g	Creeping Mimic
h	Hydra
i	Giant Insect
j	Jackal
k	Kobold
l	Giant Louse
m	Mage
n	Naga
o	Orc
p	Person
q	Quadruped
r	Rodent
s	Skeleton
t	Troll
u	Minor Demon
v	Vortex
w	Worm
x	Xorn
y	Yeek
z	Zombie
A	Angel
B	Bird
C	Canine
D	Ancient Dragon

Elemental	Floating Eye
Gelatinous Cube	Gargoyle
Ghost	Ghoul
Hybrid	Hydroid
Insect	Intelligent Thing
Jelly	Jellyfish
Kobold	Kobold Beetle
Lich	Lizard/Ent
Mold	Multi-Headed Hydra
Naga	Naga (Cursed)
Ogre	Ogre
Plant	Plant “person”
Quadruped	Quadruped (Pulsing Flesh Mound)
Reptile	Reptile/Amphibian
Spider/Scorpion	Spider/Scorpion/Tick
Townsperson	Townsperson
Major Demon	**Major Demon**
Vampire	Vampire
Wight/Wraith	Wight/Wraith/Worm Mass
Xorn	Xorn/Xaren
Yeti	Yeti
Zombie/Mummy	Zombie/Mummy

The Town Level

The town level is where you will begin your adventure. The town consists of eight buildings (each with an entrance), some townspeople, and a wall which surrounds the town and may contain streams of lava. The first time you are in town it will be daytime, but note that the sun rises and falls (rather instantly) as time passes.

Townspeople

The town contains many different kinds of people. There are the street urchins, young children who will mob an adventurer for money, and seem to come out of the woodwork when excited. Blubbering idiots are a constant annoyance, but not harmful. Public drunks wander about the town singing, and are of no threat to

anyone. Sneaky rogues hang about watching for a likely victim to mug. And finally, no town would be complete without a swarm of half drunk warriors, who take offense or become annoyed just for the fun of it. (There are assumed to be other people in the town, but they are not represented on the screen as they do not interact with the player in any way.)

Most of the townspeople should be avoided by the largest possible distance when you wander from store to store. Fights will break out, though, so be prepared. Since your character grew up in this world of intrigue, no experience is awarded for killing the town inhabitants, though you may acquire treasure.

Town Buildings

Your character will begin their adventure with some basic supplies, and some extra gold with which to purchase more supplies at the town stores.

You may enter any open store to buy items of the appropriate type. The price the shopkeeper requests is dependent on the price of the item. By default stores will not buy items from the player. If you choose to play with selling enabled, stores have a maximum value; they will not pay more than that for any item, regardless of how much it is actually worth.

Once inside a store, you will see the name and race of the store owner, the name of the store, the maximum amount of cash that the store owner will pay for any one item, and the store inventory, listed along with the prices.

You will also see an (incomplete) list of available commands. Note that many of the commands which work in the dungeon work in the stores as well, but some do not, especially those which involve “using” objects.

Stores do not always have everything in stock. As the game progresses, they may get new items so check from time to time. Stores restock after 10000 game turns have passed, but the inventory will never change while you are in town, even if you save

the game and return. You must be in the dungeon for the store to restock. Also, if you sell them an item, it may get sold to a customer while you are adventuring, so don't always expect to be able to get back everything you have sold. If you have a lot of spare gold, you can purchase every item in a store, which will induce the store owner to bring out new stock, and perhaps even retire.

Store owners will not accept known harmful or useless items. If an object is unidentified, they will (if selling is enabled) pay you some base price for it. Once they have bought it they will immediately identify the object. If it is a good object, they will add it to their inventory. If it was a bad bargain, they simply throw the item away. You can use this feature to learn item flavors.

The General Store (1)
: The General Store sells foods, some clothing, torches, oil, shovels and picks. All of these items and some others can be sold back to the general store for money. The general store restocks like every store, but the inventory types never change.

The Armoury (2)
: The Armoury is where the town's armour is fashioned. All sorts of protective gear may be bought and sold here. The deeper into the dungeon you progress the more exotic the equipment you will find stocked in the armoury. However, some armour types will never appear here unless you sell them.

The Weaponsmith's Shop (3)
: The Weaponsmith's Shop is where the town's weapons are fashioned. Hand and missile weapons may be purchased and sold here, along with arrows, bolts, and shots. As with the armoury, not all weapon types will be stocked here, unless they are sold to the shop by the player first.

The Bookseller (4)
: The Bookseller holds supplies of the simpler books needed by magic users, and will buy the more advanced books which can be found in the dungeon.

The Alchemy shop (5)
: The Alchemy Shop deals in all types of potions and scrolls.

The Magic User's Shop (6)
: The Magic User's Shop deals in all sorts of rings, wands, amulets, and staves.

The Black Market (7)
: The Black Market will sell and buy anything at extortionate prices. However it occasionally has **very** good items in it. With the exception of artifacts, every item found in the dungeon may appear in the black market.

Your Home (8)
: This is your house where you can store objects that you cannot carry on your travels, or will need at a later date.

Within The Dungeon

Once your character is adequately supplied with food, light, armor, and weapons, they are ready to enter the dungeon. Move on top of the > symbol and use the "Down" command (>).

Your character will enter a maze of interconnecting staircases and finally arrive somewhere on the first level of the dungeon. Each level of the dungeon is fifty feet high (thus dungeon level "Lev 1" is often called "50 ft"), and is divided into (large) rectangular regions (several times larger than the screen) by permanent rock. Once you leave a level by a staircase, you will never again find your way back to that region of that level, but there are an infinite number of other regions at that same "depth" that you can explore later. Monsters, of course, can use the stairs, and you may eventually encounter them again, but they will not chase you up or down stairs.

In the dungeon, there are many things to find, but your character must survive many horrible and challenging encounters to find the treasure lying about.

There are two sources for light once inside the dungeon. Permanent light which has been magically placed within rooms, and a light

source carried by the player. If neither is present, the character will be unable to see. This will affect searching, picking locks, disarming traps, reading scrolls, casting spells, browsing books, etc. So be very careful not to run out of light!

A character must wield a torch or lamp in order to supply his own light. A torch or lamp burns fuel as it is used, and once it is out of fuel, it stops supplying light. You will be warned as the light approaches this point. You may use the “Fuel” command (`F`) to refuel your lantern (with flasks of oil), and it is a good idea to carry extra torches or flasks of oil, as appropriate. There are rumours of objects of exceptional power which glow with their own never-ending light.

These last two paragraphs apply to most classes, but not to necromancers. Necromancers dislike light, and shroud themselves in darkness. They are usually better off not carrying a light, but also do not gain any of the bonuses that may come from magical light sources.

Objects Found In The Dungeon

The mines are full of objects just waiting to be picked up and used. How did they get there? Well, the main source for useful items are all the foolish adventurers that proceeded into the dungeon before you. They get killed, and the helpful creatures scatter the various treasure throughout the dungeon.

Several objects may occupy a given floor location, which may or may not also contain one creature. However, doors, rubble, traps, and staircases cannot coexist with items. As below, any item may actually be a “pile” of up to 40 identical items. With the right choice of “options”, you may be able to “stack” several items in the same grid.

You pick up objects by moving on top of them. You can carry up to 23 different items in your backpack while wearing and wielding up to 12 others. Although you are limited to 23 different items, each item may actually be a “pile” of up to 40 similar items. If you `t`ake off an item, it will go into your backpack if there is room: if there is

no room in your backpack, it will drop onto the floor, so be careful when swapping one wielded weapon or worn piece of armor for another when your pack is full.

You are, however, limited in the total amount of weight that you can carry. If you exceed this value, you become slower, making it easier for monsters to chase you. Note that there is no upper bound on how much you can carry, if you do not mind being slow. Your weight “limit” is determined by your strength.

Many objects found within the dungeon have special commands for their use. Wands must be Aimed, staves must be Used, scrolls must be Read, and potions must be Quaffed. You may, in general, not only use items in your pack, but also items on the ground, if you are standing on top of them. At the beginning of the game all items are assigned a random ‘flavor’. For example potions of ‘cure light wounds’ could be ‘red potions’. If you have never used, sold, or bought one of these potions, you will only see the flavor. You can learn what type of item it is by selling it to a store, or using it (although learning by use does not always apply to magic devices). Lastly, items in stores that you have not yet identified the flavor of will be labeled ‘{unseen}’.

Chests are complex objects, containing traps, locks, and possibly treasure or other objects inside them once they are opened. Many of the commands that apply to traps or doors also apply to chests and, like traps and doors, these commands do not work if you are carrying the chest.

One item in particular will be discussed here. The scroll of “Word of Recall” can be found within the dungeon, or bought at the alchemist in town. All classes start with one of these scrolls in their inventory. It acts in two manners, depending upon your current location. If read within the dungeon, it will teleport you back to town. If read in town, it will teleport you back down to the deepest level of the dungeon which your character has previously been on. This makes the scroll very useful for getting back to the deeper levels of Angband. Once the scroll has been read it takes a while for the spell to act, so don’t expect it to save you in a crisis. During this time the word ‘recall’ will appear on the bottom of the screen below the dungeon. Reading a second scroll before the first takes effect

will cancel the action.

You may “inscribe” any object with a textual inscription of your choice. These inscriptions are not limited in length, though you may not be able to see the whole inscription on the item. The game applies special meaning to inscriptions containing any text of the form ‘@#’ or ‘@x#’ or ‘!x’ or ‘!*’, see ‘customize.txt’.

The game provides some “fake” inscriptions to help you keep track of your possessions. Weapons, armor and jewellery which have properties you don’t know about yet will get a ‘{??}’ label. Wands, staves and rods can get a ‘{tried}’ label after use, particularly if they have an effect on a monster and were tested in the absence of monsters.

It is rumored that rings of power and extra rare spell books may be found deeper in the dungeon…

And lastly, a final warning: not all objects are what they seem. The line between tasty food and a poisonous mushroom is a fine one, and sometimes a chest full of treasure will grow teeth in its lid and bite your hand off…

Cursed Objects

Some objects, often objects of great power, have been cursed. There are many curses in the game, and they can appear on any wearable object. Curses may have a negative (or sometimes positive) effect on an object’s properties, or cause bad things to happen to the player at random.

You can choose to wear the object in spite of its curses, or attempt to uncurse it using magic. A warning: failed uncursing leads to the object becoming fragile, and a fragile object may be destroyed on future curse removal attempts. It is up to you to balance the risks and rewards in your use of cursed items.

Mining

Some treasure within the dungeon can be found only by mining it

out of the walls. Many rich strikes exist within each level, but must be found and mined. Quartz veins are the richest, yielding the most metals and gems, but magma veins will have some hoards hidden within.

Mining is rather difficult without a pick or shovel. Picks and shovels have an additional magical ability expressed as '(+#)'. The higher the number, the better the magical digging ability of the tool. A pick or shovel also has plusses to hit and damage, and can be used as a weapon, because, in fact, it is one.

When a vein of quartz or magma is located, the character may wield his pick or shovel and begin digging out a section. When that section is removed, he can locate another section of the vein and begin the process again. Since granite rock is much harder to dig through, it is much faster to follow the vein exactly and dig around the granite. Eventually, it becomes easier to simply kill monsters and discover items in the dungeon to sell, than to walk around digging for treasure. But, early on, mineral veins can be a wonderful source of easy treasure.

If the character has a scroll, staff, or spell of treasure location, they can immediately locate all strikes of treasure within a vein shown on the screen. This makes mining much easier and more profitable.

Note that a character with high strength and/or a heavy weapon does not need a shovel/pick to dig, but even the strongest character will benefit from a pick if trying to dig through a granite wall.

It is sometimes possible to get a character trapped within the dungeon by using various magical spells and items. So it can be a good idea to always carry some kind of digging tool, even when you are not planning on tunneling for treasure.

There are rumors of certain incredibly profitable rooms buried deep in the dungeon and completely surrounded by permanent rock and granite walls, requiring a digging implement or magical means to enter. The same rumors imply that these rooms are guarded by incredibly powerful monsters, so beware!

Traps

There are many traps located in the dungeon of varying danger. These traps are hidden from sight and are triggered only when your character walks over them. If you have found a trap you can attempt to Disarm it, but failure may mean activating it. Traps can be physical dangers such as pits, or magical runes or inscriptions which will cause an effect when triggered. Your character may be better at disarming one of these types of traps than the other.

All characters have a chance to notice traps when they first come into view (dependent on searching skill). Some players will also get access to magical means of detecting all traps within a certain radius. If you cast one of these spells, there will be a 'Dtrap' green label on the bottom of the screen, below the dungeon map.

Some monsters have the ability to create new traps on the level that may be hidden, even if the player is in a detected zone. The detection only finds the traps that exist at the time of detection, it does not inform you of new ones that have since been created.

Staircases, Secret Doors, Passages, and Rooms

Staircases are the manner in which you get deeper or climb out of the dungeon. The symbols for the up and down staircases are the same as the commands to use them. A < represents an up staircase and a > represents a down staircase. You must move your character over the staircase before you can use it.

Most levels have at least one up staircase and at least two down staircases. You may have trouble finding some well hidden secret doors, or you may have to dig through obstructions to get to them, but you can always find the stairs if you look hard enough. Stairs, like permanent rock, and shop entrances, cannot be destroyed by any means.

Many secret doors are used within the dungeon to confuse and demoralize adventurers foolish enough to enter, although all secret

doors can be discovered by stepping adjacent to them. Secret doors will sometimes hide rooms or corridors, or even entire sections of that level of the dungeon. Sometimes they simply hide small empty closets or even dead ends. Secret doors always look like granite walls, just like traps always look like normal floors.

Creatures in the dungeon will generally know and use these secret doors, and can often be counted on to leave them open behind them when they pass through.

Level and object feelings

Unless you have disabled the option to get feelings you will get a message upon entering a dungeon giving you a general feel of how dangerous that level is.

The possible messages are :

1	This seems a quiet, peaceful place”
2	This seems a tame, sheltered place”
3	This place seems reasonably safe”
4	This place does not seem too risky”
5	You feel nervous about this place”
6	You feel anxious about this place”
7	This place seems terribly dangerous”
8	This place seems murderous”
9	Omens of death haunt this place”

This feeling depends only on the monsters present in the dungeon when you first enter it. It will not get reduced to safer feeling as you kill monsters neither will it increase if new ones are summoned. This feeling also depends on your current dungeon depth. A dungeon you feel nervous about at 2000’ is way more dangerous than a murderous one at 50’.

Once you have explored a certain amount of the dungeon you will also get a feeling about how good are the objects lying on the floor of the dungeon.

The possible messages are :

1	there is naught but cobwebs here.”
2	there are only scraps of junk here.”
3	there aren’t many treasures here.”
4	there may not be much interesting here.”
5	there may be something worthwhile here.”
6	there are good treasures here.”
7	there are very good treasures here.”
8	there are excellent treasures here.”
9	there are superb treasures here.”
$	you sense an item of wondrous power!”

The last message indicates an artifact is present and is only possible if the preserve option is disabled (if preserve is enabled, an artifact will guarantee a feeling of 5 or better).

You may review your level feeling any time by using the ^K command. You may also consult it by checking the LF: indicator at the bottom left of the screen. The first number after it is the level feeling and the second one is the object feeling. The second one will be ? if you need to explore more before getting a feeling about the value of the treasures present in the dungeon.

Winning The Game

If your character has killed Sauron (a difficult task), who lives on level 99 (4950’) in the dungeon, a magical staircase will appear that will allow you to finally reach level 100. Morgoth lurks on this level of his dungeon, and you will not be able to go below his level until you have killed him. Try to avoid wandering around on level 100 unless you are ready for him, since he has a habit of coming at you across the dungeon, the Mighty Hammer ‘Grond’ in hand, to slay you for your impudence.

If you should actually survive the attempt of killing Morgoth, you will receive the status of WINNER. You may continue to explore, and may even save the game and play more later, but since you have defeated the toughest creature alive, there is really not much point. Unless you wish to listen to the rumors of a powerful ring buried somewhere in the dungeon, or a suit of dragon scale mail that resists everything…

When you are ready to retire, simply kill your character (using the Q key) to have your character entered into the high score list as a winner. Note that until you retire, you can still be killed, so you may want to retire before wandering into yet another horde of greater demons.

Upon Death and Dying

If your character falls below 0 hit points, they have died and cannot be restored. A tombstone showing information about your character will be displayed. You are also permitted to get a record of your character, and all your equipment (identified) either on the screen or in a file.

Your character will leave behind a reduced save file, which contains only your option choices. It may be restored, in which case a new character is generated exactly as if the file was not there.

There are a variety of ways to “cheat” death (including using a special “cheating option”) when it would otherwise occur. This will fully heal your character, returning him to the town, and marking him in various ways as a character which has cheated death. Cheating death, like using any of the “cheating options”, will prevent your character from appearing on the high score list.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Attacking monsters

Attacking and Being Attacked

Attacking is simple in Angband. If you move into a creature, you attack it. You can attack from a distance by firing a missile or by magical means (such as aiming a wand). Creatures attack in the same way. If they move into you, they attack you. Some creatures can also cast spells from a distance, and others can use various breath weapons (such as fire) on you from a distance.

Creatures in walls can not be attacked by wands or other magic attacks normally stopped by walls, nor can they be shot at with bows and arrows. Tunnelling into the wall (using the “tunnel” or “alter” command) will allow you to attack any creature in the wall with your main weapon. This applies to creatures which “pass through” walls: if they “bore through” walls, the wall is no longer there, and the creature can be targeted normally.

If you are wielding a weapon, the damage for the weapon is used when you hit a creature. Otherwise you get a single punch which does minimal damage.

You may wield one weapon for melee combat, and also one missile launcher (bow, crossbow or sling). You may also wear one amulet (around the one and only neck of the character), two rings (on the two “ring” fingers, i.e. the third finger of each hand: a magic ring does not function when worn on any other finger, nor may two be worn on the same finger), one light source, and a full set of armor - body armor, shield, helmet, gloves, boots and a cloak. Any or all of these items may provide powers to the character in terms of

bonuses to-hit, to-damage, to-armor class, or to other stats.

Firing a missile (while wielding the appropriate launcher) is the only way to get the “full” power out of the missile. You may of course throw an arrow at a monster without shooting it, but you will find the effects will not be what you had hoped.

Hits and misses are determined by ability to hit versus armor class. A hit is a strike that does some damage; a miss may in fact reach a target, but fails to do any damage. Higher armor classes make it harder to do damage, and so lead to more misses. Characters with higher armor classes also receive a damage reduction. This is not true for monsters, whose AC only affects the character’s difficulty to hit them.

If you wish to see how much damage your weapon will do, you can `I`nspect it. You will find the number of blows and how much damage you would do per round, including information on whether your weapon damages other types of monsters differently.

Monster Memories

There are hundreds of different creatures in the pits of Angband, many of which have the same letter symbol and color on the screen. The exact species of a creature can be discovered by `l`ooking at it. It is also very difficult to keep track of the capabilities of various creatures. Luckily, Angband automatically keeps track of your experiences with a particular creature. This feature is called the monster memory. Your monster memory recalls the particular attacks of each creature (whether or not technically a monster) which you have suffered, as well as recalling if you have observed them to multiply or move erratically, or drop treasure, etc. Otherwise you would simply have to take notes, which is an unnecessary bother.

If you have killed enough of a particular creature, or suffered enough attacks, recalling the monster memory may also provide you with information not otherwise available, such as an armor class, hit dice, spell types, frequency of spell casting, or the amount of damage for breaths or spells. These attacks will be color coded to

inform you of whether or not you currently resist a specific attack. Red or orange means you do not resist it, yellow means you partially resist it, and green means you resist it or are immune. If you attack a monster with specific elemental attacks you will learn if the monster resists that element or if they are immune. There are other magical means to learn about monster's abilities that don't require you to actually experience the attacks.

This memory can be used by all your characters; it is stored in a file called 'lore.txt' in your user directory (~/.angband/Angband in Linux, lib/user in Windows, Documents/Angband in macOS).

Your Weapon

Carrying a weapon in your backpack does you no good. You must wield a weapon before it can be used in a fight. A secondary weapon can be kept by keeping it in the backpack, and switching it with the primary weapon when needed. This is most often used when switching between two weapons, each of which provides a rare power that the character needs at two separate times. Note that a digging tool need only be carried in your pack, as when you try to dig your best digging tool will automatically be used.

Weapons have two main magical characteristics, their enchanted ability to hit and their enchanted ability to do damage, expressed as '(+#,+#)'. A normal weapon would be '(+0,+0)'. Many weapons in Angband have bonuses to hit and/or to damage.

Angband assumes that your youth in the rough environment near the dungeons has taught you the relative merits of different weapons, and displays as part of their description the damage dice which define their capabilities. Any damage enchantment is added to the dice roll for that weapon. The dice used for a given weapon is displayed as 'XdY'. The number `X` indicates how many dice to roll, and number `Y` indicates how many sides they have. A '2d6' weapon will thus give damage from 2 to 12, plus any damage bonus. The weight of a weapon is also a consideration. Heavy weapons may hit harder, but they are also harder to use. Depending on your strength, dexterity, character class, and weapon weight, you may get attacks more quickly: high dexterity and strength and low weapon weight

are the main factors. Warriors may get up to a maximum of 6 attacks per round: pure spellcasters are limited to only 4: other classes may get up to 5. Your attacks per round with a weapon are displayed as a decimal, e.g. 2.3 or 3.4 etc. The fractions take the form of unused energy which is carried over to your next turn.

Missile weapons, such as bows, have their characteristics added to those of the missile used, if the proper weapon/missile combination is used, and then the launcher multiplier is applied to the total damage, making missile weapons very powerful given the proper missiles, especially if they are enchanted. Like weapons, `I`nspecting ammunition will tell you how much damage you will do with your current missile launcher.

Finally, some rare weapons have special abilities. These are called ego weapons, and are feared by great and meek. An ego weapon must be wielded to receive the benefit of its abilities. It should be noted that some of these items are considerably more powerful than others, and generally the most powerful items are the rarest. Some items will have an obvious effect, like an increase in infravision, or extra strength. These effects will be noticed as soon as you wield the item. Other effects, like most resistances, will need to be learned. You can learn them by either suffering an appropriate attack, or by using magical means of identification.

Some of the more common ego weapons are described at the end of this file.

Your Armor Class

Your armor class (or AC) is a number that describes the amount and the quality of armor being worn. Armor class will generally run from about 0 to 200, though exceptionally good armor can improve even on the latter figure.

The higher your armor class, the more protective it is. A negative armor class would actually help get you hit. Armor protects you in three manners. First, it makes you harder to be hit for damage. A hit for no damage counts as a miss, and is described as a miss. Second, good armor will absorb some of the damage that your

character would have taken from normal attacks. Third, acid damage is reduced by wearing body armor (but the armor may be damaged instead). It is obvious that a high armor class is vital for surviving the deeper levels of Angband.

Armor class values are always displayed between a set of square brackets, as '[#]' or '[#,+#]'. The first value is the base armor class of the armor. The second number is the magical bonus of the item, which is only displayed if known, and will always have a sign preceding the value. These plusses can be determined by wielding the armor in combat and being hit. Note that a few rings, amulets, and weapons also have the '[+#]' notation, indicating that they provide an armor bonus. Many pieces of heavy body armor will also have a '(-#)' (in normal brackets) before the '[#,+#]', which indicates that the weight of the armor decreases your chances of hitting monsters. This can range from nonexistent for very light armor to '(-8)' for the heaviest armor!

Monster status effects

You will find some spells and items which can affect monsters in ways which do not involve directly dealing them damage. These are 'status effects'. They are listed with their effects below. These status effects will either work on a monster type or they won't; some monsters resist particular effects but not all do.

Hold Monster:
: Paralyses a monster until you hit them. Increases chance of player getting a critical hit. Normal duration 3-8 turns.

Stun Monster:
: Reduces the monster's melee accuracy and damage by 25%. 1 in 10 chance that the monster will miss the turn. Increases chance of player getting a critical hit. Normal duration 5-10 turns.

Confuse Monster:
: Monster spells fail 50% more often. Monster at least 40% more likely to miss target with spells/ranged attacks. Monster ball & bolt spells sometimes go in the wrong direction. 30%

chance of erratic movement, more when more confused. Increases chance of player getting a critical hit. Normal duration 5-10 turns.

Slow Monster:
: -2 speed, more if more slowed. Normal duration 10 or more turns.

Sleep Monster:
: Puts monsters to sleep, but they can wake up again quite easily.

Scare Monster:
: Monster will run away. Monster spells fail 20% more often.

Disenchant Monster:
: Monster spells fail 50% more often. Normal duration 5-10 turns.

Non-melee attacks and resistances

The player may at some time gain access to non-melee attacks, and many monsters also have them. Perhaps the most famous of this type of attack is dragon breath, but monsters may also cast spells at the player, and vice versa. This damage generally is not affected by armor class, and does not need a hit roll to hit the player or monster being aimed at.

Some attacks are purely magical: attack spells which blind, confuse, slow, scare or paralyze the target. These attacks are resisted by monsters of higher level (native to deeper dungeon depths) and characters with a high saving throw - saving throws being dependent on class, level and wisdom. There are also available resistances to fear, blindness, confusion and stunning, and the power of “free action” prevents magical paralysis and most slowing attacks (the player may still be paralyzed by being “knocked out” in melee or by a stunning attack, but this is very rare and can be prevented with protection from stunning.) There are monsters that can cause status effects such as blindness, paralysis or confusion through their melee attack. Since this is a physical effect and not a

mental one, the player will not get a saving throw. However, having resistance to that effect will prevent the negative status in all cases. It should also be noticed that most unique monsters automatically pass their saving throws, and some monsters are naturally resistant to confusion, fear and sleep. Some monsters may have spells that 'cause wounds' that can be deadly if successful but do no damage if the saving throw is passed. Some melee attacks by monsters may drain a stat, as can some traps: this is prevented by having that stat sustained. Drained stats are temporary and can be restored on gaining a new character level or consuming rare items found in the dungeon.

Some monsters may cast spells that teleport the player character. There is no saving throw, except to those that would actually teleport him up or down one dungeon level. Having resistance to nexus will also prevent being level-teleported, but will not help against normal teleportation spell attacks. The player may teleport monsters in the same way, with a spell, wand or rod. No monsters, even Morgoth himself, can resist this teleportation. Yet…

Other attacks are usually element-based, including the aforementioned example of dragon breath. Many monsters can breathe various attacks or cast bolt or ball spells, and the player may also have access to bolt and ball spells (or breathe like a dragon, in some rare circumstances). The player, and the monsters, may be resistant to these forms of attack: resistance is handled in different ways for the player and the monster, and for different attack forms.

Bolt spells will hit the first monster (or the player) in the line of fire: ball spells may centre on a target which may be hiding behind other targets. Ball spells and breath weapons affect an area: other monsters caught in the blast take reduced damage depending on their distance from the centre of the blast. Breath weapons are proportional to a fraction of the monster's current hit points and drop off in power with distance from the monster, with a maximum cap on the damage (which is higher for the most common of such attacks, owing to the fact that the resistances are also easier to find). Bolt and ball spell damage is calculated differently - often (but not always) relative to character or monster level.

In the case of fire, cold, lightning, acid and poison, if the monster has resistance to a player attack of this kind it will take almost no damage. If the player has one or more permanent sources of resistance, they will take 1/3 of the damage they would normally take: if the player has a temporary source of resistance (whether from potion, spell or item activation), this will also reduce the damage to 1/3 of its normal level, allowing the character to take only 1/9 damage if they have both permanent and temporary resistance. Having more than one source of permanent resistance confers no extra bonus, and using more than one source of temporary resistance increases only the duration of the resistance: in both cases, either the resistance is present or it is not. But one permanent resistance and one temporary resistance are both effective simultaneously.

Elemental attacks also have a chance to damage wielded equipment or destroy items in the character's inventory. Fire attacks destroy scrolls, staves, magic books and arrows. Acid attacks destroy scrolls, staves, arrows, bolts and can damage armor. Electricity attacks can destroy wands, rods, rings and amulets. Cold attacks can destroy potions. Items in your inventory get a saving throw, and they are unharmed if they pass it. Having resistance to the element will make an item less likely to be destroyed. Items on the floor that get caught in an elemental ball or breath are automatically destroyed without a saving throw. Weapons, armor and chests can also be destroyed if they are lying on the floor, but cannot be harmed if they are in your pack.

The character may also gain immunity to fire, cold, lightning and acid if he is fortunate to find any of the few artifacts that provide these immunities: immunity means that no damage is taken, and the character's equipment is also totally protected. Immunities are EXTREMELY rare.

Another attack that the player will come into contact with all too often is the soul-chilling nature of the undead, which can drain the character's life experience. Some monsters have a life-draining melee attack, others may cast ball or bolt spells or, in extreme cases, breathe the very force of the netherworld (shortened by the game to "nether".) There are two powers which are of assistance in

this case: that of “hold life” will prevent 90% of all experience drains, and in the other 10% of cases, the amount of experience lost will be reduced by 90%. That of “resistance to nether forces” will provide resistance to nether bolts, balls and breaths, reducing the damage and preventing any experience drains from these attacks, but has no effect on melee “hits to drain experience”. Monsters caught in the blast from a nether ball or breath will take damage proportional to distance from the centre of the attack, except for undead who are totally immune. The player may find wands or rods of Drain Life, which similarly are ineffective on those undead creatures which have no life to drain: however, the real player equivalent attack spell is the priest/paladin spell of “Orb of Draining”, a ball spell which does damage to all monsters, double damage to evil monsters, and is resisted by none.

Other attack forms are rarer, but may include: disenchantment (both in melee or by a monster breath), chaos (breath or melee, which if unresisted will cause the player to hallucinate and be confused, and may drain life experience), nexus (which may teleport the player to the monster, away from the monster, up or down a level, or swap over two of the player’s “internal” stats), light and darkness (which will blind a character unless they have protection from blindness or resistance to light or dark), sound (which will stun a character without sound resistance or protection from stunning), crystal shards (which will cut a non-resistant character), inertia (which will slow a character regardless of free action), gravity (which will blink a character, also stunning and slowing), force (which will stun the character), plasma (which will stun), time (which may drain experience regardless of hold life, or drain stats regardless of sustains), water bolts and balls (which may confuse and stun, and do considerable damage from high-level monsters), ice bolts (which may cut and stun, and damage potions), and mana bolts and balls (the latter usually known as Mana Storms.) Magic missiles are included in the “mana” category, whether cast by the monster or the player.

In addition items on the ground are especially vulnerable to elemental effects. Potions on the ground will always be destroyed by cold, shards, sound and force. Scrolls, staves, books, and non-metal gear will always get destroyed by fire or plasma. Scrolls,

staves, and all non-mithril gear will be destroyed by acid. Rings, amulets, wands and rods will be destroyed by lightning and plasma. And finally nearly everything will be destroyed by a mana storm if left on the ground.

Some attacks may stun or cut the player. These can either be spells or breath attacks (sound, water balls) or from melee. A stunned character receives a penalty to hit and is much more likely to fail a spell or activation. If a character gets very stunned, they may be knocked out and at the mercy of the enemies. A cut character will slowly lose life until healed either by potions, spells or natural regeneration. Both stunning and cut status are displayed at the bottom of the screen.

There are resistances available to chaos, disenchantment, confusion, nexus, sound, shards, light and darkness: all of these will reduce the damage and prevent side-effects other than physical damage. With these resistances, as with nether resistance, damage is a random fraction between 1/2 and 2/3.

It should be noted that not all of these are actually vital to completing the game: indeed, of the above list, only fire, cold, acid, lightning, poison and confusion resists are regarded as truly vital, with blindness, chaos and nether the next most desirable. Some attack forms are not resistible, but thankfully these are rare: resist shards will prevent all other magical attacks which cut (namely ice bolts), and confusion resistance will prevent confusion by a water bolt or ball, but there is no resistance to the physical damage caused by these following attacks: inertia, force, gravity, plasma, time, ice, water, mana. There is no resistance to any of the side-effects of a time attack, or indeed to anything but the stunning effects of a gravity attack.

A note on speed

Monsters which do not move at normal speed generally move “slowly” (-10 to speed), “fairly quickly” (+5), “quickly” (+10), “very quickly” (+20) or “incredibly quickly” (+30). (It will surprise nobody that Morgoth is one of the few monsters in the last category.) This is further adjusted by the fact that any non-unique

monster may have a random adjustment from (-2) to (+2) to its own speed.

Generally, (+10) is exactly double normal speed, and (-10) exactly half. (+20) is about three times normal speed, but after that there is less noticeable improvement as speed goes higher - for instance, (+30) is not quite four times normal speed, and higher values than this are largely irrelevant. The player may find items which can be worn or wielded that provide speed bonuses: these may include boots of speed, rings of speed and a few very rare artifacts. Boots will provide a random 1d10 to speed: rings of speed may be bigger than that - generally the best that the player will get is two just over (+10), but individual rings of up to (+23) speed have been known.

Separate from the question of permanent speed (as determined by the player's speed items and the monster's natural speed) is that of temporary speed. The player may cast a spell of haste-self, or use a potion, staff or rod of speed or use an artifact activation to speed him temporarily: or a monster may cast a haste-self spell, or be affected by another monster "shrieking for help" or the player reading a scroll of aggravate monster. In all cases, (+10) speed is added temporarily to the affected monster or player. Using two or more sources of temporary speed is cumulative only in duration - one cannot get from normal speed to (+20) using a potion and a spell of speed. Spells of temporary slowing (including monsters breathing inertia or gravity) are handled the same way, with exactly (-10) being subtracted from the player or monster's speed temporarily, for the duration of the spell or breath's effect.

Ego weapons and armor

Some of the ego weapons that you might find in the dungeon are listed below. This will give you a small taste of the items that can be found. However if you wish to discover these items on your own, you may not wish to continue. Ego weapons are denoted by the following "names":

Ego Melee Weapons:

(Defender)

A magical weapon that actually helps the wielder defend himself, thus increasing his/her armor class, and protecting him/her against damage from fire, cold, acid, lightning, and falls. This weapon also will increase your stealth, let you see invisible creatures, protect you from paralyzation and some slowing attacks, and help you regenerate hit points and mana faster. As a result of the regeneration ability, you will use up food somewhat faster than normal while wielding such a weapon. These powerful weapons also will sustain one stat, though this stat will vary from weapon to weapon.

(Holy Avenger)

A Holy Avenger is often one of the most powerful weapons. A Holy Avenger will increase your wisdom and your armour class. This weapon will do extra damage when used against evil, demonic and undead creatures, and will also give you the ability to see invisible creatures. These weapons are basically extremely powerful versions of Blessed Blades and can be wielded by priests with no penalty. These weapons, like (Defender) weapons, also will sustain one random stat.

(Blessed)

A blessed blade will increase your wisdom. If you are a priest, wielding a non-blessed sword or polearm causes a small penalty while attacking and may infuriate your god, decreasing the chances that she will accept your prayers: a blessed blade may be wielded without this penalty. Blessed blades also have one extra, random, power.

Weapon of Westernesse

A Weapon of Westernesse is one of the more powerful weapons. It does extra damage against orcs, trolls, and giants, while increasing your strength, dexterity, and constitution. It also lets you see invisible creatures and protects from paralyzation and some slowing attacks. These blades were made by the Dunedain.

Weapon of Extra Attacks

A weapon of extra attacks will allow the wielder to deliver extra attacks during each round.

Elemental Branded Weapons

Each of the five elemental attacks has a corresponding weapon which will do treble its base damage to creatures not resistant to that element. (It should be noted that the magical damage bonus is not affected by this: a weapon of Flame '(2d6) (+5, +6)' does 6d6+6 damage per hit, not 6d6+18, against creatures which are not fire-resistant.) There are weapons of Flame, Frost, Lightning, Acid and Poison brands.

Weapons of Slaying enemies

These weapons do extra damage against creatures of a vulnerable type. Weapons of Slay Evil and Slay Animal do double the base damage, while weapons of Slay Orc, Troll, Giant, Dragon, Demon and Undead do triple the base damage. As with elemental branded weapons, the magical damage bonus is not affected.

Weapons of *Slay*ing enemies

These weapons, in addition to doing extra damage to your enemies, have extra powers as well. In each case, one stat is increased. Weapons of *Slay* Dragon, Demon or Undead are also more powerful against their opponents, doing five times their base damage rather than the normal three.

Shovels and Picks of Digging

These powerful diggers will dig through granite as if it were mere wood, and mineral veins as if they were butter. Permanent rock is still an impassable obstacle.

Ego Missile Launchers and Ammo:

Launchers of Accuracy

These launchers have an unnaturally high to-hit number, making them extremely accurate.

Launchers of Power

These launchers do an unnaturally high amount of damage due to their high to-dam number.

Launchers of Extra Shots

These launchers allow the wielder to shoot more times per

round than normal.

Launchers of Extra Might
: These launchers have a higher base damage than normally made launchers of their type. For instance, a 'Long Bow of Extra Might (x3)(+X,+Y)(+1)' is really a Long Bow '(x4)(+X,+Y)' where '(+X,+Y)' is the standard to-hit and to-dam. As the damage multiplier with the bow affects **everything** the base arrow damage, the magical damage bonus on both the bow and the arrow, and any bonuses for slaying or elemental-branded arrows - this makes it a powerful weapon.

Ammo of Wounding
: This ammunition - whether it be pebbles, iron shots, arrows, bolts, seeker arrows or seeker bolts - has big bonuses to-hit and to-damage.

Ammo of Elemental Brands, and Ammo of Slaying enemies
: This works in the same way as melee weapons of the same type: double damage for slay evil and slay animal, triple damage for all other slays and for all elemental brands. Unlike melee weapons, the slays and elemental brands **do** affect the magical damage bonus for ammo.

These are the most common types of ego-weapon: note that they are not the ONLY ego-items available in the dungeon, there may be more.

Apart from these there are some very rare and well made weapons in the dungeon with not necessarily any special abilities. These include Blades of Chaos, Maces of Disruption, and Scythes of Slicing. They can also be ego weapons like the ones above. For example, a Blade of Chaos (Holy Avenger) is much more powerful than many artifact weapons!

Some pieces of armor will possess special abilities denoted by the following names:

Ego Armors and Shields:

of Resist Acid, Lightning, Fire or Cold
: A character wearing armor or a shield with one such resistance will take only 1/3 of normal damage from attacks involving the relevant element of acid, lightning, fire or cold. Note that multiple permanent sources of resistance are NOT cumulative: wearing two is no better than wearing one. However, armor which provides resistance to acid cannot itself be damaged by acid, and this is a good reason to wear more than one such piece of armor.

of Resistance
: A character wearing armor with this ability will have resistance to Acid, Cold, Fire, and Lightning as explained in each part above.

Armor of Elvenkind
: This is the same as Resistance armor, only generally better enchanted. It will make you more stealthy. This armor also possesses an extra resistance, at random from the following list: poison, light, dark, nexus, nether, chaos, disenchantment, sound, and shards.

Robes of Permanence
: These robes are designed especially for wizards. Just like Elvenkind armor, they provide resistance to fire, cold, acid, and electricity and cannot be damaged by acid. They sustain all of your stats and protect you from a good deal of all experience draining. Also like Elvenkind armor, they have one random resistance.

Dragon Scale Mails
: These extremely rare pieces of armour come in many different colors, each protecting you against the relevant dragons. Naturally they are all resistant to acid damage. They also occasionally allow you to breathe as a dragon would. Dragon Scale Mails can also have egos as well.

Ego Helms:

Stat Boosting Helms

There are magical helms found in the dungeon that have the ability to boost the wearer's intelligence or wisdom. In addition to boosting the relevant stat these helms will also prevent that stat from being drained.

Crown of the Magi

This is the great crown of the wizards. The wearer will have an increased (and sustained) intelligence, and will also be given resistance against fire, frost, acid, and lightning. These valuable helms also have an additional random power.

Crown of Might

This is the crown of the warriors. The wearer will have an increased and sustained strength, dexterity, and constitution, and will also be immune to any foe's attempt to slow or paralyze him or her.

Crown of Lordliness

This is the great crown of the priests. The wearer will have an increased and sustained wisdom.

Helm/Crown of Seeing

This is the great helmet or crown of the rogues. The wearer will be able to see invisible creatures, and will have an increased ability to locate traps. It is also rumored that the wearer of such a helm will not be able to be blinded.

Helm of Infravision

This helmet allows the character to see monsters even in total darkness, with the ability to see heat. Note that spellbooks are the same temperature as the surroundings, and so cannot be read unless some real light is present. (Some monsters which are invisible to normal vision can be seen under infravision.)

Helm of Light

In addition to providing a permanent light source for the wearer, this helm also provides resistance against light-based attacks.

Helm/Crown of Telepathy

This helm or crown grants the wearer the power of telepathy.

Helm of Regeneration

This helm will help you regenerate hit points and mana more quickly than normal, allowing you to fight longer before needing to rest. You will use food faster than normal while wearing this helm because of the regenerative effects.

Ego Cloaks:

Cloak of Protection

This finely made cloak will come with an unnaturally high enchantment and is not affected by elemental based attacks.

Cloak of Stealth

This cloak will increase the wearer's stealth, making the wearer less likely to wake up sleeping monsters.

Cloak of Aman

These exceptionally rare cloaks provide great stealth, have a very high enchantment, and one random resistance.

Ego Gloves:

Gloves of Free Action

The wearer of these gloves will find himself resistant to paralyzing attacks as well as some slowing attacks. Because of the special nature of these gloves, magic users may wear these gloves without incurring a mana penalty.

Gloves of Slaying

These gloves will increase the wearer's fighting ability by boosting the wearer's to-hit and to-dam values.

Gloves of Agility

These gloves will increase the wearer's dexterity. Because of the special nature of these gloves, magic users may wear these gloves without incurring a mana penalty.

Gauntlets of Power

These spiked gauntlets will boost the wearer's strength as well as the wearer's to-hit and to-dam numbers.

Ego Boots:

Boots of Slow Descent
: These boots protect the wearer from the effects of small falls.

Boots of Stealth
: These boots increase the wearer's stealth, like a Cloak of Stealth.

Boots of Free Action
: The wearer of these boots will find himself resistant to paralyzing attacks as well as some slowing attacks.

Boots of Speed
: The wearer of these boots will become unnaturally fast.

Once again, these are not necessarily the ONLY ego-items in the dungeon, only the most common.

Apart from these there are some very rare and well-made armours in the dungeon with not necessarily any special abilities. These include Shields of Deflection, Adamantite Plate Mail, Mithril Plate Mail, Mithril Chain Mail, and Elven Cloaks. The first four cannot be damaged by acid because of the quality metals they contain.

There are rumors of unique "artifact" items in the dungeon - weapons and armor of all types. Many of these are more powerful than even the greatest ego-items: some are weak and have little more than a name to recommend them.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Playing the Game

Most of your interaction with Angband will take the form of “commands”. Every Angband command consists of an “underlying command” plus a variety of optional or required arguments, such as a repeat count, a direction, or the index of an inventory object. Commands are normally specified by typing a series of keypresses, from which the underlying command is extracted, along with any encoded arguments. You may choose how the standard “keyboard keys” are mapped to the “underlying commands” by choosing one of the two standard “keysets”, the “original” keyset or the “roguelike” keyset.

The original keyset is very similar to the “underlying” command set, with a few additions (such as the ability to use the numeric “directions” to “walk” or the `5` key to “stay still”). The roguelike keyset provides similar additions, and also allows the use of the `h/j/k/l/y/u/b/n` keys to “walk” (or, in combination with the shift or control keys, to run or alter), which thus requires a variety of key mappings to allow access to the underlying commands used for walking/running/altering. In particular, the “roguelike” keyset includes many more “capital” and “control” keys, as shown below.

Note that any keys that are not required for access to the underlying command set may be used by the user to extend the “keyset” which is being used, by defining new “keymaps”. To avoid the use of any “keymaps”, press backslash (`\`) plus the “underlying command” key. You may enter “control-keys” as a caret (`^`) plus the key (so `^ + p` yields ‘^p’).

Some commands allow an optional “repeat count”, which allows

you to tell the game that you wish to do the command multiple times, unless you press a key or are otherwise disturbed. To enter a "repeat count", type `0`, followed by the numerical count, followed by the command. You must type 'space' before entering certain commands. Skipping the numerical count yields a count of 99. An option allows certain commands (open, disarm, alter, etc) to auto-repeat.

Some commands will prompt for extra information, such as a direction, an inventory or equipment item, a spell, a textual inscription, the symbol of a monster race, a sub-command, a verification, an amount of time, a quantity, a file name, or various other things. Normally you can hit return to choose the "default" response, or escape to cancel the command entirely.

Some commands will prompt for a spell or an inventory item. Pressing space (or `*`) will give you a list of choices. Pressing `-` (minus) selects the item on the floor. Pressing a lowercase letter selects the given item. Pressing a capital letter selects the given item after verification. Pressing a numeric digit `#` selects the first item (if any) whose inscription contains '@#' or '@x#', where `x` is the current "underlying command". You may only specify items which are "legal" for the command. Whenever an item inscription contains '!*' or '!x' (with `x` as above) you must verify its selection.

Some commands will prompt for a direction. You may enter a "compass" direction using any of the "direction keys" shown below. Sometimes, you may specify that you wish to use the current "target", by pressing `t` or `5`, or that you wish to select a new target, by pressing `*` (see "Target" below).

Original Keyset Directions

Roguelike Keyset Directions

Each of the standard keysets provides some short-cuts over the "underlying commands". For example, both keysets allow you to "walk" by simply pressing an "original" direction key (or a "roguelike" direction key if you are using the roguelike keyset), instead of using the "walk" command plus a direction. The roguelike keyset allows you to "run" or "alter" by simply holding the shift or control modifier key down while pressing a "roguelike" direction key, instead of using the "run" or "alter" command plus a direction. Both keysets allow the use of the `5` key to "stand still", which is most convenient when using the original keyset.

Original Keyset Command Summary

Activate an object	Activate an object
(unused)	Browse a book
Close a door	Close a door
Disarm a trap or lock a door	Display character sheet
Eat an item	Disarm a trap or lock a door
Fire quiver	Eat an item
Fire your lantern/torch	Fire quiver
Get objects off floor	Fire your lantern/torch
Hurl object/spell/prayer	Get objects off floor
Inspect contents of pack	Hurl object/spell/prayer
(unused)	Fire default ammo at target
Look at item	Inspect contents of pack
Look at player	(unused)
Display map	Toggle ignore
Display map of entire level	Look at player
Repeat previous command	Display map
Open door or chest	Display map of entire level
(unused)	Repeat previous command
Quaff a potion	Open door or chest
Read a scroll	(unused)
Search for a period	Quaff a potion
Steal (rogues only)	Read a scroll
Take off equipment	Search for a period
Use a staff	Steal (rogues only)
Display version info	Take off equipment
Walk/wield equipment	Use a staff
(unused)	Display version info
(unused)	Walk/wield equipment
Zap a rod	(unused)
	(unused)
	Zap a rod

(reserved - debug command)
(unused)
(reserved - break)
(unused)
Toggle inven/equip window
(special control key)
(unused)
(unused)
Target monster or location
(reserved - tab)
(special control key)
(unused)
Describe an object
Describe an object
Display visible monster list
Display visible object list
Show previous message
Show previous messages
(reserved)
Redraw the screen
Walk and pickup
(unused)
Target closest monster
Enter user pref command
(reserved - wizard mode)
Save and quit
(unused)
(unused)
(special keypress)
(special keypress)
Identify symbol
List contents of quiver

Roguelike Keyset Command Summary

Activate/Activate
(walk southwest)
Display character sheet
Disarm trap or lock a door
List equipped items
Fuel lantern/torch

Get objects/spells/prayers	
(walk west)	
Inspect item of pack	
(walk south)	
(walk north)	
(walk east)	
Display map of entire level	
(walk southeast)	
Toggle ignore for chest	
Browse a book	
Quit character & quit	
Rest for a period	
Steal (rogues only)	
Take off equipment	
(walk northeast)	
Display version info	
Wear/wield equipment (Where)	
Use an item	
(walk northwest)	
Aim a wand (Zap)	
(special debug command)	
Enter wall (phase)	
(special break)	
Ignore an item	
Toggle inven/equip window	
(special control key)	
Do not pickup	
Target monster or location	
(special tab)	
Dump screen to a file	
Inscribe an object	
Unwield an object	
Display visible monster list	
Display visible object list	
Show previous message	
Show previous messages	
Aim wand	
Set options	
Walk (and pickup)	
Take notes	

T	Target closest monster
r	Repeat previous command
(	(special - wizard mode)
S	Save and quit
s	Search (with pickup)
)	(special - wizard mode)
<	Ascend staircase
\	(special - bypass keymap)
!	(special - shell escape)
/	Identify symbol
f	Fire default ammo at target
l	List contents of quiver

Special Keys

Certain special keys may be intercepted by the operating system or the host machine, causing unexpected results. In general, these special keys are control keys, and often, you can disable their special effects.

If you are playing on a UNIX or similar system, then ‘Ctrl-c’ will interrupt Angband. The second and third interrupt will induce a warning bell, and the fourth will induce both a warning bell and a special message, since the fifth will quit the game, after killing your character. Also, ‘Ctrl-z’ will suspend the game, and return you to the original command shell, until you resume the game with the ‘fg’ command. There is now a compilation option to force the game to prevent the “double ‘ctrl-z’ escape death trick”. The ‘Ctrl-\’ and ‘Ctrl-d’ and ‘Ctrl-s’ keys should not be intercepted.

It is often possible to specify “control-keys” without actually pressing the control key, by typing a caret (^) followed by the key. This is useful for specifying control-key commands which might be caught by the operating system as explained above.

Pressing backslash (\) before a command will bypass all keymaps, and the next keypress will be interpreted as an “underlying command” key, unless it is a caret (^), in which case the keypress after that will be turned into a control-key and interpreted as a command in the underlying Angband keyset. The backslash key is useful for creating actions which are not affected by any keymap

definitions that may be in force, for example, the sequence `\ + . + 6` will always mean “run east”, even if the `.` key has been mapped to a different underlying command.

The `0` and `^` and `\` keys all have special meaning when entered at the command prompt, and there is no “useful” way to specify any of them as an “underlying command”, which is okay, since they would have no effect.

For many input requests or queries, the special character ‘ESCAPE’ will abort the command. The ‘[y/n]’ prompts may be answered with `y` or `n`, or ‘escape’. The ‘-more-’ message prompts may be cleared (after reading the displayed message) by pressing ‘ESCAPE’, ‘SPACE’, ‘RETURN’, ‘LINEFEED’, or by any keypress, if the ‘quick_messages’ option is turned on.

Command Counts

Some commands can be executed a fixed number of times by preceding them with a count. Counted commands will execute until the count expires, until you type any character, or until something significant happens, such as being attacked. Thus, a counted command doesn’t work to attack another creature. While the command is being repeated, the number of times left to be repeated will flash by on the line at the bottom of the screen.

To give a count to a command, type 0, the repeat count, and then the command. If you want to give a movement command and you are using the original command set (where the movement commands are digits), press space after the count and you will be prompted for the command.

Counted commands are very useful for time consuming commands, as they automatically terminate on success, or if you are attacked. You may also terminate any counted command (or resting or running), by typing any character. This character is ignored, but it is safest to use a ‘SPACE’ or ‘ESCAPE’ which are always ignored as commands in case you type the command just after the count expires.

You can tell Angband to automatically use a repeat count of 99 with commands you normally want to repeat (open, disarm, tunnel, bash, alter, etc) by setting the 'always_repeat' option.

Selection of Objects

Many commands will also prompt for a particular object to be used. For example, the command to read a scroll will ask you which of the scrolls that you are carrying that you wish to read. In such cases, the selection is made by typing a letter of the alphabet (or a number if choosing from the quiver). The prompt will indicate the possible letters/numbers, and you will also be shown a list of the appropriate items. Often you will be able to press / to switch between inventory and equipment, or | to select the quiver, or – to select the floor. Using the right arrow also rotates selection between equipment, inventory, quiver, floor and back to equipment; the left arrow rotates in the opposite direction.

The particular object may be selected by an upper case or a lower case letter. If lower case is used, the selection takes place immediately. If upper case is used, then the particular option is described, and you are given the option of confirming or retracting that choice. Upper case selection is thus safer, but requires an extra key stroke.

Shape Changes

Some classes, objects, or races may allow your character to change shape: becoming, for instance, a fox or a wolf. While in the alternate shape, your character will not have access to items in the pack or quiver and will not be able to access items on the floor except for eating or pickup. The items your character was wearing upon changing shape will remain equipped and continue to affect the character's statistics, resistances, number of blows, and damage. Your character will not be able to activate any equipped items while in the alternate shape. To have your character change back to normal, cast a spell or use one of the commands, like drop, that uses an item.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Command Descriptions

The following command descriptions are listed as the command name plus the default key to use it. For those who prefer the original “roguelike” keyset, the name and key of the roguelike command is also shown if it is different. Then comes a brief description of the command, including information about alternative methods of specifying the command in each keyset, when needed.

Some commands use the “repeat count” to automatically repeat the command several times, while others use the “repeat count” to specify a “quantity” for the command, and still others use it as an “argument” of some kind.

Most commands take no “energy” to perform, while other commands only take energy when they cause the world to change in some way. For example, attempting to read a scroll while blind does not use any energy.

The following command is very useful for beginners,

Command lists (‘Enter’)
: This brings up a little window in the middle of the screen, in which you can select what command you would like to use by browsing. If you wish to begin playing immediately, you can use this option to navigate the commands and refer to this guide when you need more details about specific commands.

Inventory Commands

Inventory list (`i`)

Displays a list of objects being carried but not equipped. You can carry up to 23 different items, not counting those in your equipment. Often, many identical objects can be “stacked” into a “pile” which will count as a single item. Each object has a weight, and if you carry more objects than your strength permits, you will begin to slow down. The amount of weight you can still carry without being overencumbered, or the amount of extra weight you are currently carrying is displayed at the top of the screen.

Equipment list (`e`)

Use this command to display a list of the objects currently being used by your character. The standard body (which all races currently have) has 12 slots for equipment. Every equipment slot corresponds to a different location on the body, and each of which may contain only one object at a time, and each of which may only contain objects of the proper “type”. For the standard body these are WEAPON (weapon), BOW (missile launcher), RING (ring) (two of these), AMULET (amulet), LIGHT (light source), BODY_ARMOR (armor), CLOAK (cloak), SHIELD (shield), HAT (helmet), GLOVES (gloves), BOOTS (boots). You must be wielding/wearing certain objects to take advantage of their special powers.

Quiver list (`|`)

Missiles that you carry will automatically be put in your quiver. The quiver has 10 slots; it also takes up inventory space, so every 40 missiles will reduce your number of inventory slots by 1.

Drop an item (`d`)

This drops an item from your inventory or equipment onto the dungeon floor. If the floor spot you are standing on already has an object in it, Angband will attempt to drop the item onto an adjacent space. Doors and traps are considered objects for the purpose of determining if the space is occupied. This command may take a quantity, and takes some energy.

Ignore an item (`k`) or Ignore an item (‘^d’)

This ignores an item in your inventory or on the dungeon floor. If the selected pile contains multiple objects, you may specify a quantity. When ignored, the game will sometimes prompt you whether to ignore only this item or all others like it. If the second option is chosen, all similar items on the floor and in your inventory will be ignored. To view all items regardless of whether they are ignored, you can use `K` to toggle the ignore setting on and off.

Wear/Wield equipment (`w`)

To wear or wield an object in your inventory, use this command. Since only one object can be in each slot at a time, if you wear or wield an item into a slot which is already occupied, the old item will be first be taken off, and may in fact be dropped if there is no room for it in your inventory. Wielding ammunition will add it to an empty slot in your quiver and prompt you to replace a type of ammunition if your quiver is already full. This command takes some energy.

Take off equipment (`t`) or Take off equipment (`T`)

Use this command to take off a piece of equipment and return it to your inventory. Occasionally, you will run into a cursed item which cannot be removed. These items normally penalize you in some way and cannot be taken off until the curse is removed. If there is no room in your inventory for the item, your pack will overflow and you will drop the item after taking it off. You may also remove ammunition from your quiver with this command. This command takes some energy.

Movement Commands

Moving (arrow keys, number keys) or (arrow keys, number keys, ‘yuhjklbn’)

This causes you to move one step in a given direction. If the square you wish to move into is occupied by a monster, you will attack it. If the square is occupied by a door or a trap you may attempt to open or disarm it if the appropriate option is set. Preceding this command with CTRL will cause you to attack in the appropriate direction, but will not move your

character if no monster is there. These commands take some energy.

Walk (W)
: The walk command lets you willingly walk into a trap or a closed door, without trying to open or disarm it. This command may take a count, requires a direction, and takes some energy.

Run (.) or Run (,)
: This command will move in the given direction, following any bends in the corridor, until you either have to make a "choice" between two directions or you are disturbed. You can configure what will disturb you by setting the disturbance options. You may also use shift plus the "roguelike" direction keys (roguelike keyset), or shift plus the "original" direction keys on the keypad (both keysets, some machines) to run in a direction. This command may take an argument, requires a direction, and takes some energy.

Go up staircase (<)
: Climbs up an up staircase you are standing on. There is always at least one staircase going up on every level except for the town level (this doesn't mean it's easy to find). Going up a staircase will take you to a new dungeon level unless you are at 50 feet (dungeon level 1), in which case you will return to the town level. Note that whenever you leave a level (not the town), you will never find it again. This means that for all intents and purposes, any objects on that level are destroyed. This includes artifacts unless the "Create characters in preserve mode" option was set when your character was created, in which case the artifacts may show up again later. This command takes some energy.

Go down staircase (>)
: Descends a down staircase you are standing on. There are always at least one staircase going down on each level, except for the town which has only one, and "quest" levels, which have none until the quest monster is killed. Going down a staircase will take you to a new dungeon level. See "Go Up Staircase" for more info. This command takes some energy.

Resting Commands

Stay still (with pickup) (`,`) or Stay still (with pickup) (`.`)
: Stays in the same square for one move. If you normally pick up objects you encounter, you will pick up whatever you are standing on. You may also use the `5` key (both keysets). This command may take a count, and takes some energy.

Get objects (`g`)
: Pick up objects and gold on the floor beneath you. Picking up gold takes no time, and objects take 1/10th of a normal turn each (maximum time cost is a full turn). You may pick up objects until the floor is empty or your backpack is full.

Rest (`R`)
: Resting is better for you than repeatedly staying still, and can be told to automatically stop after a certain amount of time, or when various conditions are met. In any case, you always wake up when anything disturbing happens, or when you press any key. To rest, enter the Rest command, followed by the number of turns you want to rest, or `*` to rest until your hitpoints and mana are restored, or `&` to rest until you are fully "healed". This command may take an argument (used for the number of turns to rest), and takes some energy.

Alter Commands

Tunnel (`T`) or Tunnel ('^t')
: Tunnelling or mining is a very useful art. There are many kinds of rock, with varying hardness, including permanent rock (permanent), granite (very hard), quartz veins (hard), magma veins (soft), and rubble (very soft). Quartz and Magma veins may be displayed in a special way, and may sometimes contain treasure, in which case they will be displayed in a different way. Rubble sometimes covers an object but is easy to tunnel through, even with your bare hands. Tunnelling ability increases with strength and weapon weight. If you have a digging tool in your pack, the game will automatically use this to dig. This command may take a

count, requires a direction, and takes some energy.

Open a door or chest (o)
: To open an object such as a door or chest, you must use this command. If the object is locked, you will attempt to pick the lock based on your disarming ability. If you open a trapped chest without disarming the traps first, the trap will be set off. Opening will automatically attempt to pick any door locks. You may need several tries to open a door or chest. This command may take a count, requires a direction, and takes some energy.

Close a door (c)
: Non-intelligent and some other creatures cannot open doors, so shutting doors can be quite valuable. Furthermore, monsters cannot see you behind closed doors, so closing doors may allow you to buy some time without being attacked. Broken doors cannot be closed. This command may take a count, requires a direction, and takes some energy.

Disarm a trap or chest, or lock a door (D)
: You can attempt to disarm traps on the floor or on chests. If you fail, there is a chance that you will blunder and set it off. You can only disarm a trap after you have found it. The command can also be used to lock a closed door, which will create a hindrance for monsters. Even if many monsters will be able to pick the lock or bash the door down, it will often take them some time. This command may take a count, requires a direction, and takes some energy.

Alter (+)
: This special command allows the use of a single keypress to select any of the “obvious” commands above (attack, tunnel, bash, open, disarm), and, by using keymaps, to combine this keypress with directions. In general, this allows the use of the “control” key plus the appropriate “direction” key (including the roguelike direction keys in roguelike mode) as a kind of generic “alter the terrain feature of an adjacent grid” command. This command may take a count, requires a direction, and takes some energy.

Steal (`s`)

This command is only available to rogues, and allows them to try and steal from a monster. Stealing works better when the player is stealthy and faster than the target monster, and best of all when the victim is asleep. A failed theft will wake the monster; if you really bungle the attempt, the monster may shout out in anger. This command requires a direction and takes some energy.

Spell Commands

Browse a book (`b`) or Peruse a book (`P`)

Each class has books it can read and books it cannot; except for warriors, who cannot read any books. When this command is used, all of the spells contained in the selected book are displayed, along with information such as their level, the amount of mana required to cast them, and whether or not you know the spell.

Gain new spells (`G`)

Use this command to actually learn new spells. When you are able to learn new spells, the word “Study” will appear on the status line at the bottom of the screen. If you have a book in your possession, containing spells which you may learn, then you may choose to study that book. Most classes may actually choose which spell to study, but if you are a priest or paladin, your gods will choose a prayer for you. There are five books of each realm, but hybrid classes - paladins, rogues, rangers and blackguards - can only cast from two or three of these. Higher level books are normally found only in the dungeon. This command takes some energy.

Cast a spell (`m` in both keysets)

To cast a spell, you must have previously learned the spell and must have in your inventory a book from which the spell can be read. Each spell has a chance of failure which starts out fairly large but decreases as you gain levels. If you don’t have enough mana to cast a spell, you will be prompted for confirmation. If you decide to go ahead, the chance of failure is greatly increased, and you may wind up paralyzed for

several turns. Since you must read the spell from a book, you cannot be blind or confused while casting, and there must be some light present. This command takes some energy: the higher your level, the less it takes, but the higher the spell level, the more it takes.

Object Manipulation Commands

Eat some food (`E`)

You must eat regularly to prevent starvation. There is a hunger meter at the bottom of the screen, which says “Fed” and gives a percentage in most circumstances. If you go hungry long enough, you will become weak, then start fainting, and eventually, you may well die of starvation (accompanied by increasingly alarming messages on your hunger meter). It is also possible to be “Full”, which will make you move slowly; more slowly the fuller you get. You may use this command to eat food in your inventory. Note that you can sometimes find food in the dungeon, but it is not always wise to eat strange food. This command takes some energy.

Fuel your lantern/torch (`F`)

If you are using a lantern and have flasks of oil in your pack, then you can “refuel” them with this command. Torches and Lanterns are limited in their maximal fuel. In general, two flasks will fully fuel a lantern. This command takes some energy.

Quaff a potion (`q`)

Use this command to drink a potion. Potions affect the player in various ways, but the effects are not always immediately obvious. This command takes some energy.

Read a scroll (`r`)

Use this command to read a scroll. Scroll spells usually have an area effect, except for a few cases where they act on other objects. Reading a scroll causes the parchment to disintegrate as the scroll takes effect. Most scrolls which prompt for more

information can be aborted (by pressing escape), which will stop reading the scroll before it disintegrates. This command takes some energy.

Inscribe an object ({)

This command inscribes a string on an object. The inscription is displayed inside curly braces after the object description. The inscription is limited to the particular object (or pile) and is not automatically transferred to all similar objects. Under certain circumstances, Angband will display “fake” inscriptions on certain objects (‘tried’, ‘empty’) when appropriate. These “fake” inscriptions remain all the time, even if the player chooses to add a “real” inscription on top of it later.

In addition, Angband will place the inscription ‘??’ on an object for you if the object has a property (or “rune”) that you have not learned yet. This inscription will remain until you know all the runes on the object.

An item labeled as ‘{empty}’ was found to be out of charges, and an item labeled as ‘{tried}’ is a “flavored” item which the character has used, but whose effects are unknown. Certain inscriptions have a meaning to the game, see ‘@#’, ‘@x#’, ‘!*’, and ‘!x’, in the section on inventory object selection.

Uninscribe an object (})

This command removes the inscription on an object. This command will have no effect on “fake” inscriptions added by the game itself.

Toggle ignore (K) or Toggle ignore (O)

This command will toggle ignore settings. If on, all ignored items will be hidden from view. If off, all items will be shown regardless of their ignore setting. See the customize section for more info.

Magical Object Commands

Activate an object (`A`)

> You have heard rumors of special weapons and armor deep in the Pits, items that can let you breathe fire like a dragon or light rooms with just a thought. Should you ever be lucky enough to find such an item, this command will let you activate its special ability. Special abilities can only be used if you are wearing or wielding the item. This command takes some energy.

Aim a wand (`a`) or Zap a wand (`z`)

> Wands must be aimed in a direction to be used. Wands are magical devices, and therefore there is a chance you will not be able to figure out how to use them if you aren't good with magical devices. They will fire a shot that affects the first object or creature encountered or fire a beam that affects anything in a given direction, depending on the wand. An obstruction such as a door or wall will generally stop the effects from traveling any farther. This command requires a direction and can use a target. This command takes some energy.

Use a staff (`u`) or Zap a staff (`Z`)

> This command will use a staff. A staff is normally very similar to a scroll, in that they normally either have an area effect or affect a specific object. Staves are magical devices, and there is a chance you will not be able to figure out how to use them. This command takes some energy.

Zap a rod (`z`) or Activate a rod (`a`)

> Rods are extremely powerful magical items, which cannot be burnt or shattered, and which can have either staff-like or wand-like effects, but unlike staves and wands, they don't have charges. Instead, they draw on the ambient magical energy to recharge themselves, and therefore can only be activated once every few turns. The recharging time varies depending on the type of rod. This command may require a direction (depending on the type of rod, and whether you are aware of its type) and can use a target. This command takes some energy.

Throwing and Missile Weapons

Fire an item (`f`) or Fire an item (`t`)

This command will allow you to fire a missile from either your quiver or your inventory provided it is the appropriate ammunition for the current missile weapon you have equipped. You may not fire an item without a missile weapon equipped. Fired ammunition has a chance of breaking. This command takes some energy.

Fire default ammo at nearest (`h`) or ('TAB')

If you have a missile weapon equipped and the appropriate ammunition in your quiver, you can use this command to fire at the nearest visible enemy. This command will cancel itself if you lack a launcher, ammunition or a visible target that is in range. The first ammunition of the correct type found in the quiver is used. This command takes some energy.

Throw an item (`v`)

You may throw any object carried by your character. Depending on the weight, it may travel across the room or drop down beside you. Only one object from a pile will be thrown at a time. Note that throwing an object will often cause it to break, so be careful! If you throw something at a creature, your chances of hitting it are determined by your plusses to hit, your ability at throwing, and the object's plusses to hit. Some weapons are especially designed for throwing. Once the creature is hit, the object may or may not do any damage to it. Note that flasks of oil will do some fire damage to a monster on impact. If you are wielding a missile launcher compatible with the object you are throwing, then you automatically use the launcher to fire the missile with much higher range, accuracy, and damage, than you would get by just throwing the missile. Throw, like fire, requires a direction. Targeting mode (see the next command) can be invoked with `*` at the 'Direction?' prompt. This command takes some energy.

Targeting Mode (`*`)

This will allow you to aim your ranged attacks at a specific

monster or grid, so that you can point directly towards that monster or grid (even if this is not a “compass” direction) when you are asked for a direction. You can set a target using this command, or you can set a new target at the “Direction?” prompt when appropriate. At the targeting prompt, you have many options. First of all, targeting mode starts targeting nearby monsters which can be reached by “projectable” spells and thrown objects. In this mode, you can press `t` (or `5` or `.`) to select the current monster, space to advance to the next monster, `-` to back up to the previous monster, direction keys to advance to a monster more or less in that direction, `r` to “recall” the current monster, `q` to exit targeting mode, and `p` (or `o`) to stop targeting monsters and enter the mode for targeting a location on the floor or in a wall. Note that if there are no nearby monsters, you will automatically enter this mode. Note that hitting `o` is just like `p`, except that the location cursor starts on the last examined monster instead of on the player. In this mode, you use the “direction” keys to move around, and the `q` key to quit, and the `t` (or `5` or `.`) key to target the cursor location. Note that targeting a location is slightly “dangerous”, as the target is maintained even if you are far away. To cancel an old target, simply hit `*` and then ‘ESCAPE’ (or `q`). Note that when you cast a spell or throw an object at the target location, the path chosen is the “optimal” path towards that location, which may or may not be the path you want. Sometimes, by clever choice of a location on the floor for your target, you may be able to convince a thrown object or cast spell to squeeze through a hole or corridor that is blocking direct access to a different grid. Launching a ball spell or breath weapon at a location in the middle of a group of monsters can often improve the effects of that attack, since ball attacks are not stopped by interposed monsters if the ball is launched at a target.

Looking Commands

Full screen map (`M`)

This command will show a map of the entire dungeon, reduced by a factor of nine, on the screen. Only the major

dungeon features will be visible because of the scale, so even some important objects may not show up on the map. This is particularly useful in locating where the stairs are relative to your current position, or for identifying unexplored areas of the dungeon.

Locate player on map (`L`) or Where is the player (`W`)
: This command lets you scroll your map around, looking at all sectors of the current dungeon level, until you press escape, at which point the map will be re-centered on the player if necessary. To scroll the map around, simply press any of the "direction" keys. The top line will display the sector location, and the offset from your current sector.

Look around (`l`) or Examine things (`x`)
: This command is used to look around at nearby monsters (to determine their type and health) and objects (to determine their type). It is also used to find out if a monster is currently inside a wall, and what is under the player. When you are looking at something, you may hit space for more details, or to advance to the next interesting monster or object, or minus (−) to go back to the previous monster or object, or a direction key to advance to the nearest interesting monster or object (if any) in that general direction, or `r` to recall information about the current monster race, or `q` or escape to stop looking around. You always start out looking at "yourself".

Inspect an item (`I`)
: This command lets you inspect an item. This will tell you things about the special powers of the object, as well as attack information for weapons. It will also tell you what resistances or abilities you have noticed for the item and if you have not yet completely identified all properties.

List visible monsters (`[`)
: This command lists all monsters that are visible to you, telling you how many there are of each kind. It also tells you whether they are asleep, and where they are (relative to you).

List visible items (`]`)

This command lists all items that are visible to you, telling you how of each there are and where they are on the level relative to your current location.

Message Commands

Repeat level feeling (‘^f’)

Repeats the feeling about the monsters in the dungeon level that you got when you first entered the level. If you have explored enough of the level, you will also get a feeling about how good the treasures are.

View previous messages (‘^p’)

This command shows you all the recent messages. You can scroll through them, or exit with ESCAPE.

Take notes (`:`)

This command allows you to take notes, which will then appear in your message list and your character history (prefixed with “Note:”).

Game Status Commands

Character Description (`C`)

Brings up a full description of your character, including your skill levels, your current and potential stats, and various other information. From this screen, you can change your name or use the file character description command to save your character status to a file. That command saves additional information, including your background, your inventory, and the contents of your house.

Check knowledge (`~`)

This command allows you to ask about the knowledge possessed by your character. Information that you can look up is:

objects

Will display which objects your character is familiar

with. For each type of object, allows you to change whether or not it is ignored, the representation of that type on the screen, or the inscription automatically applied to all objects of that type. Some types of objects your character will be familiar with from the start of the game. Others come in “flavors”, and your character must determine the effect of each “flavor” once for each such type of object. For a type of object with a known “flavor”, you be also be able to display a summary of what the object can do.

runes
: Will display the “runes”, properties of enchanted objects, your character is familiar with. Allows you to change the inscription that is automatically appended to an object that has the rune. Once your character identifies a “rune” on one object, he or she will recognize that property on other objects.

artifacts
: Will display all artifacts that your character has encountered. Normally, once an artifact is “generated” and “lost”, it can never again be found, and will become “known” to the player. With “preserve” mode, an artifact can never be “lost” until it is “known” to the player. In either case, any “known” artifacts not in the possession of the player will never again be “generated”.

ego items
: Will display the “egos” your character has encountered. Each “ego” is a collection of enchantments that can appear on an object. “Egos” are often restricted to only a few specific types of objects.

monsters
: Displays the kinds of monsters your current or previous characters have encountered. For each kind of monster, allows you to change its representation on the screen. Some monsters are “uniques” which can be only be killed once per game. For a “unique” that your current

or previous characters have encountered, this will display whether that “unique” is still alive in this game.

features
: Displays the types of map grids that can appear in the game. For each type, allows you to change its representation on the screen and how that representation changes depending on the amount of light present.

traps
: Displays the types of traps that can appear in the game. For each type, allows you to change its representation on the screen and how that representation changes depending on the amount of light present.

shapechange effects
: Provides a more detailed description of the “shapes”, magical effects from some spells and a few items which change the shape of your character’s body.

stores and home
: Each of these will display the contents of the corresponding store or your player’s home at the time your character last visited the town. If your character is currently in town, what is displayed here will be the current contents.

hall of fame
: Displays a list of current and past characters, sorted by how far they progressed.

character history
: Displays a summary of what your current character has done.

equippable comparison
: This displays a summary of the known properties of the equippable items your character has access to, whether they are currently equipped, in your character’s pack, on the floor at your character’s current location, or in a

store. Near the top of the display is a line, beginning with "@", which summarizes the state of your character given his or her current equipment. Every line after that corresponds to an item, sorted by which equipment slot it can fill. The first character on each of those lines is the representation of that item as it would appear in the map if it was on the floor. After that is single character, "e" for equipped, "p" for pack, "f" for floor, "h" for home, and "s" for store, which indicates where the item is. The remainder of the line summarizes the properties of the object, with one property per column. In the default view, those properties are the resistances, flags, and modifiers present on the item; they appear in the same order (left to right) as they appear (top to bottom and then left to right) in the second part of the character description. You can toggle back and forth between that view and one that displays the effect of each item on your character's key statistics by pressing 'v'. You can use 'c' to cycle through which items, based on their location, are included in the display. The default is to show only the items that are equipped, in the pack, on the floor at your character's current location, and in the home. The other options are: show only the items in stores other than the home, show all items, or show only those that equipped or in the pack. There are some additional commands, notably for filtering which items are displayed based on a particular property and for displaying the details about one or two items. To see what those additional commands are, use the '?' key to bring up the in-game help for the equippable comparison.

Saving and Exiting Commands

Save and Quit ('Ctrl-x')

To save your game so that you can return to it later, use this command. Save files will also be generated (hopefully) if the game crashes due to a system error. After you die, you can use your savefile to play again with the same options and

such.

Save (‘Ctrl-s’)
: This command saves the game but doesn’t exit Angband. Use this frequently if you are paranoid about having your computer crash (or your power go out) while you are playing.

Quit (Q)
: Kills your character and exits Angband. You will be prompted to make sure you really want to do this, and then asked to verify that choice. Note that dead characters are dead forever.

User Pref File Commands

Interact with options (=)
: Allow you to interact with options. Note that using the “cheat” options may mark your savefile as unsuitable for the high score list. The “window” options allow you to specify what should be drawn in any of the special sub-windows (not available on all platforms). See the help files ‘customize.txt’ and ‘options.txt’ for more info. You can also interact with keymaps under this menu.

Interact with keymaps - option submenu
: Allow you to interact with keymaps. You may load or save keymaps from user pref files, or define keymaps. You must define a “current action”, shown at the bottom of the screen, before you attempt to use any of the “create macro” commands, which use that “current action” as their action.

Interact with visuals - option submenu
: Allow you to interact with visuals. You may load or save visuals from user pref files, or modify the attr/char mappings for the monsters, objects, and terrain features. You must use the “redraw” command (‘^r’) to redraw the map after changing attr/char mappings. NOTE: It is generally easier to modify visuals via the “knowledge” menus.

Interact with colors - option submenu
: Allow the user to interact with colors. This command only

works on some systems. NOTE: It is commonly used to brighten the 'Light Dark' color (eg. Cave Spiders) on displays with bad alpha settings.

Help Commands

Help (?)

Brings up the Angband on-line help system. Note that the help files are just text files in a particular format, and that other help files may be available on the Net. In particular, there are a variety of spoiler files which do not come with the standard distribution. Check the place you got Angband from or ask on the Angband forums, angband.oook.cz , about them.

Identify Symbol (/)

Use this command to find out what a character stands for. For instance, by pressing '/.', you can find out that the . symbol stands for a floor spot. When used with a symbol that represents creatures, the this command will tell you only what class of creature the symbol stands for, not give you specific information about a creature you can see. To get that, use the Look command.

There are three special symbols you can use with the Identify Symbol command to access specific parts of your monster memory. Typing 'Ctrl-a' when asked for a symbol will recall details about all monsters, typing 'Ctrl-u' will recall details about all unique monsters, and typing 'Ctrl-n' will recall details about all non-unique monsters.

If the character stands for a creature, you are asked if you want to recall details. If you answer yes, information about the creatures you have encountered with that symbol is shown in the Recall window if available, or on the screen if not. You can also answer k to see the list sorted by number of kills, or p to see the list sorted by dungeon level the monster is normally found on. Pressing 'ESCAPE' at any point will exit this command.

Game Version (V)

This command will tell you what version of Angband you are using. For more information, see the 'version.txt' help file.

Extra Commands

Toggle Choice Window ('^e')

Toggles the display in any sub-windows (if available) which are displaying your inventory or equipment.

Redraw Screen ('^r')

This command adapts to various changes in global options, and redraws all of the windows. It is normally only necessary in abnormal situations, such as after changing the visual attr/char mappings, or enabling "graphics" mode.

Save screen dump ( ))

This command dumps a "snap-shot" of the current screen to a file, including encoded color information. The command has two variants:

- html, suitable for viewing in a web browser.
- forum embedded html for vBulletin, suitable for pasting in web forums like http://angband.oook.cz/forums.

Special Keys

Certain special keys may be intercepted by the operating system or the host machine, causing unexpected results. In general, these special keys are control keys, and often, you can disable their special effects.

If you are playing on a UNIX or similar system, then Ctrl-c will interrupt Angband. The second and third interrupt will induce a warning bell, and the fourth will induce both a warning bell and a special message, since the fifth will quit the game, after killing your character. Also, 'Ctrl-z' will suspend the game, and return you to the original command shell, until you resume the game with the 'fg' command. There is now a compilation option to force the game to

prevent the “double ‘ctrl-z’ escape death trick”. The ‘Ctrl-\’ and ‘Ctrl-d’ and ‘Ctrl-s’ keys should not be intercepted.

It is often possible to specify “control-keys” without actually pressing the control key, by typing a caret (^) followed by the key. This is useful for specifying control-key commands which might be caught by the operating system as explained above.

Pressing backslash (\) before a command will bypass all keymaps, and the next keypress will be interpreted as an “underlying command” key, unless it is a caret (^), in which case the keypress after that will be turned into a control-key and interpreted as a command in the underlying angband keyset. For example, the sequence \ + . + 6 will always mean “run east”, even if the . key has been mapped to a different underlying command.

The 0 and ^ and \ keys all have special meaning when entered at the command prompt, and there is no “useful” way to specify any of them as an “underlying command”, which is okay, since they would have no effect.

For many input requests or queries, the special character ESCAPE will abort the command. The ‘[y/n]’ prompts may be answered with y or n, or ‘ESCAPE’. The ‘-more-’ message prompts may be cleared (after reading the displayed message) by pressing ‘ESCAPE’, ‘SPACE’, ‘RETURN’, ‘LINEFEED’, or by any keypress, if the “quick_messages” option is turned on.

Command Counts

Some commands can be executed a fixed number of times by preceding them with a count. Counted commands will execute until the count expires, until you type any character, or until something significant happens, such as being attacked. Thus, a counted command doesn’t work to attack another creature. While the command is being repeated, the number of times left to be repeated will flash by on the line at the bottom of the screen.

To give a count to a command, type 0, the repeat count, and then the command. If you want to give a movement command and you

are using the original command set (where the movement commands are digits), press space after the count and you will be prompted for the command.

Counted commands are very useful for time consuming commands, as they automatically terminate on success, or if you are attacked. You may also terminate any counted command (or resting or running), by typing any character. This character is ignored, but it is safest to use a ‘SPACE’ or ‘ESCAPE’ which are always ignored as commands in case you type the command just after the count expires.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Option Descriptions

Options are accessible through the = command, which provides an interface to the various sets of options available to the player.

In the descriptions below, each option is listed as the textual summary which is shown on the “options” screen, plus the internal name of the option in brackets, followed by a textual description of the option.

Various concepts are mentioned in the descriptions below, including “disturb”, (cancel any running, resting, or repeated commands, which are in progress), “flush” (forget any keypresses waiting in the keypress queue), “fresh” (dump any pending output to the screen), and “sub-windows” (see below).

Option pages

User Interface Options

When setting the user interface options, there are a handful of commands to make it easier to get to a well-known state for all of those options. They are: ‘s’ to save the current selections so that they will be used as the starting point for future characters, ‘r’ to

reset the current selections to the defaults for a new character, and ‘x’ to reset the current selections to the Angband maintainer’s defaults for the user interface options.

Rogue-like commands `rogue_like_commands`
: Selects the “roguelike” command set. See Playing the Game for some description of the command sets. The “roguelike” command set may work better for you if use a keyboard which doesn’t have a numeric keypad.

Use sound `use_sound`
: Turns on sound effects, if your system supports them.

Show damage player deals to monsters `show_damage`
: Shows the damage that the player deals to monsters for melee and ranged combat in the messages.

Use old target by default `use_old_target`
: Forces all commands which normally ask for a “direction” to use the current “target” if there is one. Use of this option can be dangerous if you target locations on the ground, unless you clear them when done.

Always pickup items `pickup_always`
: Automatically picks up items when you walk upon them, provided it is safe to do so.

Always pickup items matching inventory `pickup_inven`
: Like `pickup_always`, but picks up an item only if it is a copy of an item that you already have in your inventory.

Show flavors in object descriptions `show_flavors`
: Display “flavors” (color or variety) in object descriptions, even for objects whose type is known. This does not affect objects in stores.

Highlight target with cursor `show_target`
: Highlights the current targeted monster with a cursor. Useful when combined with “use old target by default”.

Highlight player with cursor between turns `highlight_player`

Highlights the player with a cursor. Useful if you have trouble finding the player.

Disturb whenever viewable monster moves `disturb_near`
Disturb the player when any viewable monster moves, whenever any monster becomes viewable for the first time, and also whenever any viewable monster becomes no longer viewable. This option ignores the existence of “telepathy” for the purpose of determining whether a monster is “viewable”.

Show walls as solid blocks `solid_walls`
Walls are solid blocks instead of # for granite and % for veins. Veins are coloured darker than granite for differentiation purposes.

Show walls with shaded backgrounds `hybrid_walls`
Walls appear as # and % symbols overlaid on a gray background block. This overrides `solid_walls`.

Color: Illuminate torchlight in yellow `view_yellow_light`
This option causes special colors to be used for “torch-lit” grids. Turning this option off will slightly improve game speed.

Color: Shimmer multi-colored things `animate_flicker`
Certain powerful monsters and items will shimmer in real time, i.e. between keypresses.

Center map continuously `center_player`
The map always centres on the player with this option on. With it off, it is divided into 25 sections, with coordinates (0,0) to (4,4), and will show one section at a time - the display will “flip” to the next section when the player nears the edge.

Color: Show unique monsters in purple `purple_uniques`
All “unique” monsters will be shown in a light purple colour, which is not used for any “normal” monsters - so you can tell at a glance that they are unique. If you like the idea but don’t like the colour, you can edit it via the “interact with colors” option.

Automatically clear -more- prompts `auto_more`
: The game does not wait for a keypress when it comes to a '-more-' prompt, but carries on going.

Color: Player color indicates low hit points `hp_changes_color`
: This option makes the player `@` turn various shades of colour from white to red, depending on percentage of HP remaining.

Allow mouse clicks to move the player `mouse_movement`
: Clicking on the main window will be interpreted as a move command to that spot.

Notify on object recharge `notify_recharge`
: This causes the game to print a message when any rechargeable object (i.e. a rod or activatable weapon or armour item) finishes recharging. It is the equivalent of inscribing '{!!}' on all such items.

Show effective speed as multiplier `effective_speed`
: Instead of showing absolute speed modifier (e.g. 'Slow (-2)' or 'Fast (+38)'), show the effective rate at which the character is moving (e.g. 'Slow (x0.8)' or 'Fast (x4.1)').

Birth options

The birth options may only be changed when creating a character or using the quick restart option for a dead character. When setting the birth options, there are a handful of commands to make it easier to get to a well-known state for all of the birth options. They are: 's' to save the current selections so that they will be used as the starting point for future characters, 'r' to reset the current selections to the defaults for a new character, and 'x' to reset the current selections to the Angband maintainer's defaults for the birth options.

Generate a new, random artifact set `birth_randarts`
: A different set of artifacts will be created, in place of the standard ones. This is intended primarily for people who have played enough to know what most of the standard artifacts do and want some variety. The number, findability and power of

random artifacts will all match the standard set - approximately.

Generate connected stairs `birth_connect_stairs`

With this option turned on, if you go down stairs, you start the new level on an up staircase, and vice versa (if you go up stairs, you start the next level on a down staircase).

With this option off, you will never start on a staircase - but other staircases up and down elsewhere on the level will still be generated.

Force player descent (never make up stairs) `birth_force_descend`

Upwards staircases do not work. All downward staircases, including the one in town, transport the character one level below the previous maximum depth. Recalling from the dungeon works and brings the character to the town. However, recalling from town brings the character one level below the previous maximum depth. The character cannot recall from quest levels until the quest is complete, however you will be warned before descending into a quest level. Any status effects that sometimes teleports the character up and sometimes teleports them down will always choose down. When combined with the option for word of recall scrolls to have no effect, this recreates the previous “ironman” option.

Word of Recall has no effect `birth_no_recall`

Word of Recall scrolls have no effect. When combined with the option to force player descent, this recreates the previous “ironman” option.

Restrict creation of artifacts `birth_no_artifacts`

No artifacts will be created. Ever. Just *how* masochistic are you?

Stack objects on the floor `birth_stacking`

With this option turned on, multiple items can occupy one grid.

With this option off, items dropped on the floor will spread out instead of stacking. Normal items will disappear if there is no empty grid within a radius of three squares.

Lose artifacts when leaving level `birth_lose_arts`

Normally if you leave a level with an unidentified artifact on it you may still find it later. With this option on, if you leave a level with an artifact on it's gone for the rest of the game whether you knew it was there or not. Note that this option has no effect on artifacts which you have already identified (i.e. picked up) - these will always be permanently lost if you leave a level without taking them with you.

Show level feelings `birth_feelings`

With this option turned on, the game will give you hints about what a new level has on it.

With this option off, these hints will not be shown.

Increase gold drops but disable selling `birth_no_selling`

Shopkeepers will never pay you for items you sell, though they will still identify unknown items for you, and will still sell you their wares. To balance out income in the game, gold found in the dungeon will be increased if this option is on.

Start with a kit of useful gear `birth_start_kit`

Start with items, a useful option for new players, or ones that wish to descend immediately into the dungeon. If turned off, the character will start with additional gold with which to purchase starting gear.

Monsters learn from their mistakes `birth_ai_learn`

Allow monsters to learn what spell attacks you are resistant to, and to use this information to choose the best attacks. This option makes the game very difficult and is not recommended.

Know all runes on birth `birth_know_runes`

For players who don't enjoy the "identify by use" process for wearable items. This option means all object properties are

known at the outset, so artifacts and ego items will be identified on walking over them.

Know all flavors on birth `birth_know_flavors`

For players who don't enjoy the "identify by use" process for consumable items. This option means all object flavors are known at the outset.

Persistent levels (experimental) `birth_levels_persist`

Each level is generated for the first time when the player enters it, and from then on when the player returns the level is as they last saw it, including monsters, items and traps.

To-damage is a percentage of dice (experimental) `birth_percent_damage`

Instead of bonuses to damage being just added on to damage dealt, each +1 adds 5% to the value of the damage dice. This option is currently not very balanced.

Cheating options

Peek into monster creation `cheat_hear`

Cheaters never win. But they can peek at monster creation.

Peek into dungeon creation `cheat_room`

Cheaters never win. But they can peek at room creation.

Peek into something else `cheat_xtra`

Cheaters never win. But they can see debugging messages.

Allow player to avoid death `cheat_live`

Cheaters never win. But they can cheat death.

Window flags

Some platforms support "sub-windows", which are windows which can be used to display useful information generally available through other means. The best thing about these windows is that they are updated automatically (usually) to reflect the current state

of the world. The “window options” can be used to specify what should be displayed in each window. The possible choices should be pretty obvious.

Display inven/equip
: Display the player inventory (and sometimes the equipment).

Display equip/inven
: Display the player equipment (and sometimes the inventory).

Display player (basic)
: Display a brief description of the character, including a breakdown of the current player “skills” (including attacks/shots per round).

Display player (extra)
: Display a special description of the character, including some of the “flags” which pertain to a character, broken down by equipment item.

Display player (compact)
: Display a brief description of the character, including a breakdown of the contributions of each equipment item to various resistances and stats.

Display map view
: Display the area around the player or around the target while targeting. This allows using graphical tiles in their original size.

Display messages
: Display the most recently generated “messages”.

Display overhead view
: Display an overhead view of the entire dungeon level.

Display monster recall
: Display a description of the monster which has been most recently attacked, targeted, or examined in some way.

Display object recall
: Display a description of the most recently selected object.

Currently this only affects spellbooks and prayerbooks. This window flag may be usefully combined with others, such as “monster recall”.

Display monster list
: Display a list of monsters you know about and their distance from you.

Display status
: Display the current status of the player, with permanent or temporary boosts, resistances and status ailments (also available on the main window).

Display item list
: Display a list of items you know about and their distance from you.

Left Over Information

The `hitpoint_warn` value, if non-zero, is the percentage of maximal hitpoints at which the player is warned that they may die. It is also used as the cut-off for using the color red to display both hitpoints and mana.

The `delay_factor` value, if non-zero, will slow down the visual effects used for missile, bolt, beam, and ball attacks. The actual time delay is equal to `delay_factor` squared, in milliseconds.

The `lazymove_delay` value, if non-zero, will allow the player to move diagonally by pressing the two appropriate arrow keys within the delay time. This may be useful particularly when using a keyboard with no numpad.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Customising the game

Angband allows you to change various aspects of the game to suit your tastes. These include:

- Options - which let you change interface or gameplay behaviour
- Ignoring items and inscribing items to change how the game treats them
- Showing extra info in subwindows
- Keymaps - a way to assign commonly-used actions to specific keys
- Visuals - allowing you to change the appearance of in-game entities like objects and monsters
- Colours - allowing you to make a given color brighter, darker, or even completely different
- Interface details - depending on which interface to the game you use, these give you control over the font, window placement, and graphical tile set

Except for the options, which are linked to the save file, and interface details, that are handled by the front end rather than the core of the game, you can save your preferences for these into files, which are called *user pref files*. For the options, customize those using the = command while playing.

User Pref Files

User pref files are Angband's way of saving and loading certain settings. They can store:

- Altered visual appearances for game entities
- Inscriptions to automatically apply to items
- Keymaps
- Altered colours
- Subwindow settings
- Colours for different types of messages
- What audio files to play for different types of messages

They are simple text files with an easy to modify format, and the game has a set of pre-existing pref files in the lib/customize/ folder. It's recommended you don't modify these.

Several options menu (=) items allow you to load existing user pref files, create new user pref files, or save to a user pref file.

Where to find them

On macOS, you can find them in your user directory, in `Documents/Angband/`.

On Linux, they will be stored in `~/.angband/Angband`.

On Windows you can find them in `lib/user/`.

How do they get loaded?

When the game starts up, after you have loaded or created a character, some user pref files are loaded automatically. These are the ones mentioned above in the `lib/customize/` folder, namely `pref.prf` followed by `font.prf`. If you have graphics turned on, then the game will also load some settings from `lib/tiles/`.

After these are complete, the game will try to load (in order):

- *race*.prf - where *race* is your character's race, so something like `Dwarf.prf`
- *class*.prf - where *class* is your character's class, so something like `Paladin.prf`
- *name*.prf - where *name* is your character's name, so something like `Balin.prf`

So, you can save some settings - for example, keymaps - to the `Mage.prf` file if you only want them to be loaded for mages.

You may also enter single user pref commands directly, using the special “Enter a user pref command” command, activated by pressing `"`.

You may have to use the redraw command (`^R`) after changing certain of the aspects of the game to allow Angband to adapt to your changes.

Ignoring items

Angband allows you to ignore specific items that you don’t want to see anymore. These items are marked ‘ignored’ and any similar items are hidden from view. The easiest way to ignore an item is with the `k` (or `^D`) command; the object is dropped and then hidden from view. When ignoring an object, you will be given a choice of ignoring just that object, or all objects like it in some way.

The entire ignoring system can also be accessed from the options menu (`=`) by choosing `i` for `Item ignoring setup`. This allows ignore settings for non-wearable items, and quality and ego ignore settings (described below) for wearable items, to be viewed or changed.

There is a quality setting for each wearable item type. Ignoring a wearable item will prompt you with a question about whether you wish to ignore all of that type of item with a certain quality setting, or of an ego type, or both.

The quality settings are:

bad
: The weapon/armor has negative AC, to-hit or to-dam.

average
: The weapon/armor has no pluses no minuses. It is non-magical.

good

The weapon/armor has positive AC, to-hit or to-dam. However it does not have any special abilities, brands, slays, stat-boosts, resistances

non-artifact
: This setting only leaves artifacts unignored.

Inscribing items

Inscriptions are notes you can mark on objects using the { command. You can use this to give the game commands about the object, which are listed below. You can also set up the game to automatically inscribe certain items whenever you find them, using the object knowledge screens, accessed using ~.

Inscribing an item with '!!':
: This will alert you when the item has finished recharging.

Inscribing an item with '=g':
: This marks an item as 'always pick up'. This is sometimes useful for picking up ammunition after a shootout. If there is a number immediately after the 'g', then the amount picked up automatically will be limited. If you have inscribed a spellbook with '=g4' and have four or more copies in your pack, you will not automatially pick up any more copies when you have the 'pickup if in inventory' option enabled. If you have three copies in your pack with that inscription and happen to find a pile of two copies, you'll automatically pick up one so there is four in the pack.

Inscribing an item with ! followed by a command letter or *:
: This means "ask me before using this item". '!w' means 'ask me before wielding', '!d' means 'ask me before dropping', and so on. If you inscribe an item with '!*' then the game will confirm any use of an item.

  Say you inscribed your potion of Speed with '!q'. This would prompt you when you try to drink it to see if you really mean to. Multiple '!q' inscriptions will prompt multiple times.

Similarly, using !v!k!d makes it very hard for you to accidentally throw, ignore or put down the item it is inscribed on.

Some adventurers use this for Scrolls of Word of Recall so they don't accidentally return to the dungeon too soon.

Inscribing an item with `@`, followed by a command letter, followed by 0-9:

Normally when you select an item from your inventory you must enter the letter that corresponds to the item. Since the order of your inventory changes as items get added and removed, this can get annoying. You can instead assign certain items numbers when using a command so that wherever they are in your backpack, you can use the same keypresses. If you have multiple items inscribed with the same thing, the game will use the first one.

For example, if you inscribe a staff of Cure Light Wounds with '@u1', you can refer to it by pressing 1 when `u`sing it. You could also inscribe a wand of Wonder with '@a1', and when using `a`, 1 would select that wand.

Spellcasters should inscribe their books, so that if they lose them they do not cast the wrong spell. If you are mage and the beginner's spellbook is the first in your inventory, casting 'maa' will cast magic missile. But if you lose your spellbook, casting 'maa' will cast the first spell in whatever new book is in the top of your inventory. This can be a waste in the best case scenario and exceedingly dangerous in the worst! By inscribing your spellbooks with '@m1', '@m2', etc., if you lose your first spellbook and attempt to cast magic missile by using 'm1a', you cannot accidentally select the wrong spellbook.

Inscribing an item with `^`, followed by a command letter:

When you wear an item inscribed with `^`, the game prompts you before doing that action. You might inscribe '^>' on an item if you want to be reminded to take it off before going down stairs. If the item is in your backpack then the game

won't prompt you.

Like with `!`, you can use `*` for the command letter if you want to game to prompt you every turn whatever you're doing. This can get very annoying!

Showing extra info in subwindows

In addition to the main window, you can create additional windows that have secondary information on them. You can access the subwindow menu by using `=` then `w`, where you can choose what to display in which window.

You may then need to make the window visible using the "window" menu from the menu bar (if you have one in your version of the game).

There are a variety of subwindow choices and you should experiment to see which ones are the most useful for you.

Keymaps

You can set up keymaps in Angband, which allow you to map a single keypress, the trigger, to a series of keypresses, the action. For example you might map the key F1 to "maa" (the keypresses to cast "Magic Missile" as a spellcaster). This can speed up access to commonly-used features. To bypass a keymap that's been assigned to a key, press `\` before pressing the key.

To set up keymaps, go to the options menu (=) and select "Edit keymaps" (`e`). There, you can check if a key triggers a keymap: select "Query a keymap" (`c`) and then press the key to check. You can also remove an existing keymap: select "Remove a keymap" (`e`) and then press the key that trigger the keymap to be removed. To add a new keymap (or overwrite an existing one), select "Create a keymap" (`d`), it will then prompt you for the key that triggers the keymap. After pressing the trigger key, you'll be prompted for the keymap's action, the series of keypresses that'll be generated when the trigger key is pressed. If you make a mistake while entering the

keypresses for the action, press `Control-u` to erase the keypresses already entered for the action. Once you've finished entering the keypresses for the action, press `=` to end the sequence; you'll then be prompted for whether to keep the newly entered keymap.

Within the action for a keymap, it is frequently useful to temporarily suppress -more- prompts since they can swallow keypresses from the keymap. To disable those prompts from within the action, include `(`. To reenable the prompts, include `)`. So, a typical action where -more- prompts could happen would look like this: `(` your keypresses here `)`.

The keypresses in the action will be interpreted relative to the keyset you are currently using (original or roguelike). The game will remember what keyset was in effect when the keymap was created. So if you change keysets, the keymaps which were only defined for the other keyset won't be visible. You can have two keymaps, one for the original keyset and another for the roguelike keyset, bound to the same trigger.

Keymaps are not recursive. If you have F1 as the trigger for a keymap, including F1 as a keypress in the action for that or another keymap won't invoke that keymap.

Any changes you make to keymaps from the options menu only last as long as the game is running. To have them affect future sessions, save the keymaps to a file. There's an option to do that from the menu for editing keymaps. See User Pref Files for how the name of the file affects whether the file is loaded when the game reloads your character.

Note that the game accounts for the modifier keys (Shift, Control, Alt, Meta) that are pressed along with a key. On most platforms, the game also distinguishes between the keys on the numeric keypad that have equivalents on the main keyboard. When a keypress is displayed or saved to the preference file, the modifiers, if any, for the keypress are displayed by code letters (S for Shift, ^ for Control, A for Alt, M for Meta, and K for the numeric keypad) within curly braces prior to the keypress. There are two exceptions to that: if Control is the only modifier it will displayed as ^ before the keypress without any curly braces and if Shift is the only modifier it

will often be folded into the keypress itself. For example:

```
{^S}& = Control-Shift-&
{AK}0 = Alt-0 from the numeric keypad
^d    = Control-d
A     = Shift-a
```

Special keys, like F1, F2, or Tab, are all written within square brackets []. For example:

```
^[F1]     = Control-F1
{^S}[Tab] = Control-Shift-Tab
```

Special keys include [Escape].

You may find it easier to edit the preference files directly to change a keymap. Keymaps are written in pref files as:

```
keymap-act:<action>
keymap-input:<type>:<trigger>
```

The action must always come first, `<type>` means 'keyset type', which is either 0 for the original keyset or 1 for the roguelike keyset. For example:

```
keymap-act:maa
keymap-input:0:[F1]
```

An action can have more than one trigger bound to it by having more than one keymap-input line after it and before the next keymap-act line. One reason to do that would be to have the keymap work with either keyset. For example:

```
keymap-act:maa
keymap-input:0:[F1]
keymap-input:1:[F1]
```

Angband uses a few built-in keymaps. These are for the movement keys (they are mapped to ; plus the number, e.g. 5 -> ;5), amongst others. You can see the full list in pref.prf, but they shouldn't impact you in any way.

Colours

The “Interact with colors” options submenu (=, then `c`) allows you to change how different colours are displayed. Depending on what kind of computer you have, this may or may not have any effect.

The interface is quite clunky. You can move through the colours using `n` for ‘next colour’ and `N` for ‘previous colour’. Then upper and lower case `r`, `g` and `b` will let you tweak the color. You can then save the results to user pref file.

Visuals

You can change how various in-game entities are displayed using the visuals editor. This editor is part of the knowledge menus (`~`). When you are looking at a particular entity - for example, a monster - if you can edit its visuals, that will be mentioned in the prompt at the bottom of the screen.

If you are in graphics mode, you will be able to select a new tile for the entity. If you are not, you will only be able to change its colours.

Once you have made edits, you can save them from the options menu (=). Press `v` for ‘save visuals’ and choose what you want to save.

Interface details

Some aspects of how the game is presented, notably the font, window placement and graphical tile set, are controlled by the front end, rather than the core of the game itself. Each front end has its own mechanism for setting those details and recording them between game sessions. Below are brief descriptions for what you can configure with the standard Windows, X11, SDL, SDL2 and Mac front ends.

Windows

With the Windows front end, the game, by default, displays several of the the subwindows and uses David Gervais's graphical tiles to display the map. You can close a subwindow with the standard close control on the window's upper right corner. Closing the main window with the standard control causes the game to save its current state and then exit. You can reopen or also close a subwindow via the "Visibilty" menu, the first entry in the "Window" menu for the main window. To move a window, use the standard procedure: position the mouse pointer on the window's title bar and then click and drag the mouse to change the window's position. Click and drag on the edges or corners of a window to change its size. To select the font for a window, use the "Font" menu, the second entry in the "Window" menu for the main window.

The "Term Options" entry in the "Window" menu for the main window is a shortcut to access the core game's method for selecting the contents of the subwindows. You can read more about that in Showing extra info in subwindows. The "Reset Layout" will rearrange the windows to conform with the current size and will have a similar result to what you would get from restarting the Windows interface without a preset configuration.

The "Bizarre Display" entry in the "Window" menu allows to toggle on or off an alternate text display algorithm for each window. That was added for compatibility with Windows Vista and later. The default setting, on, should likely be used, unless text display is garbled on your system and the off setting allows text to be displayed properly.

The "Increase Tile Width" and "Decrease Tile Width" options in the "Window", let you increment or decrement, by one pixel, the width of the columns in a window. The "Increase Tile Height" and "Decrease Tile Height" options are similar but work with the height of the rows. For the primary window, you could use the "Term 0 Font Tile Size" entry as an alternative to those to set the width of the columns and height of the rows to certain combinations or to match the width and height of the font, which is the default. When the "Enable Nice Graphics" option is on (it's in the "Options" menu for the main window), the "Increase Tile Width", "Decrease Tile

Width", "Increase Tile Height", "Decrease Tile Height", and "Term 0 Font Tile Size" entries will have no effect since the column width and row height are set automatically when that option is on.

To change whether graphical tiles are used, use the "Graphics" menu, the first entry in the "Options" menu for the main window. The "None" option in the "Graphics" menu will disable graphical tiles and use text for the map. The next section section in that menu allows you to select one of the graphical tile sets. Turning on the "Enable Nice Graphics" option in the "Graphics" menu is a shortcut for automatically setting sizes to get a reasonable-looking result. When that is turned on or is already on and the tile set is changed, the width of the columns ("tile width"), height of the rows ("tile height") and the number of rows and columns used to display a tile (the "Tile Multiplier") will be adjusted to work well with the current font size and the native size of the graphical tiles. You can manually adjust the number of rows and columns used for displaying a tile with the "Tile Multiplier" entry in the "Graphics" menu. Since typical fonts are often twice as tall as wide, multipliers where the first value, for the width, is twice the second, often x work better with the tiles that are natively square (the original ones, Adam Bolt's, David Gervais's, and the two versions of Shockbolt's tiles). Nomad's tiles are 8 x 16 and so usually work best with multipliers that use the same value for both dimensions.

When you leave the game, the current settings for the Windows interface are saved as `angband.INI` in the directory that holds the executable. Those settings will be automatically reloaded the next time you start the Windows interface.

X11

With the X11 front end, the number of windows opened is set by the '-n' option on the command line, i.e. running `./angband -mx11 -- -n4` will open the main window and subwindows one through three if the executable is in the current working directory. To control the font, placement, and size used for each of the windows, set enviroment variables before running Angband. Those environment variables for window 'z' where 'z' is an integer between 0 (the main window) and 7 are:

- ANGBAND_X11_FONT_z holds the name of the font to use for the window
- ANGBAND_X11_AT_X_z holds the horizontal coordinate (zero is leftmost) for the upper left corner of the window
- ANGBAND_X11_AT_Y_z holds the vertical coordinate (zero is topmost) for the upper left corner of the window
- ANGBAND_X11_COLS_z holds the number of columns to display in the window
- ANGBAND_X11_ROWS_z holds the number of rows to display in the window

SDL

With the SDL front end, the main window and any subwindows are displayed within the application's rectangular window. At the top of the application's window is a status line. Within that status line, items highlighted in yellow are buttons that can be pressed to initiate an action. From left to right they are:

- The application's version number - pressing it displays an information dialog about the application
- The currently selected terminal - pressing it brings up a menu for selecting the current terminal; you can also make a terminal the current one by clicking on the terminal's title bar if it is visible
- Whether or not the current terminal is visible - pressing it for any terminal that is not the main window will allow you to show or hide that terminal
- The font for the current terminal - pressing it brings up a menu to choose the font for the terminal
- Options - brings up a dialog for selecting global options including those for the graphical tile set used and whether fullscreen mode is enabled
- Quit - to save the game and exit

To move a terminal window, click on its title bar and then drag the mouse. To resize a terminal window, position the mouse pointer over the lower right corner. That should cause a blue square to appear, then click and drag to resize the terminal.

To change the graphical tile set used when displaying the game's map, press the Options button in the status bar. Then, in the dialog that appears, press one of the red buttons that appear to the right of the label, "Available Graphics:". The last of those buttons, labeled "None", selects text as the method for displaying the map. Your choice for the graphical tile set does not take effect until you press the red button labeled "OK" at the bottom of the dialog.

When you leave the game, the current settings for the SDL interface are saved as `sdlinit.txt` in the same directory as is used for preference files, see User Pref Files for details. Those settings will be automatically reloaded the next time you start the SDL interface.

SDL2

With the SDL2 front end, the application has one window that can contain the main window and any of the subwindows. The application may also have up to three additional windows which can contain any of the subwindows. A subwindow may not appear in more than of those application windows. Unused portions of an application window are tiled with repetitions of the game's logo.

Each of the application windows has a menu bar along the top. The "Menu" entry at the left end of the menu bar has the main menu for controlling aspects of the SDL2 interface.

Next to "Menu", are a series of one letter labels that act as toggles for the terminal windows shown in the application window. Click on one to toggle it between on (drawn in white) and off (drawn in gray). It is not possible to toggle off the main window shown in the primary application window.

At the end of the menu bar are two toggle buttons labeled "Size" and "Move". Each will be gray if disabled or white if enabled. Clicking on "Size" when it is disabled will enable it, disable "Move", turn off input to the game's core, and cause clicks and drags within the displayed subwindows to change the sizes for those subwindows. Clicking on "Move" when it is disabled will enable it, disable "Size", turn off input to the game's core, and cause clicks and drags within the displayed subwindows to change the positions for those subwindows. Disable both "Move" and "Size", by clicking

on one if it is enabled, to restore passing input to the game's core.

Within "Menu", the first entries control properties each of the displayed terminal windows within that application window. For the main window, you can set the font, graphical tile set, whether the window is shown with borders or not, and whether or not the window will be shown on top of the other windows. For subwindows, you can set the font, the purpose (which is a shortcut for enabling the subwindow content as described in Showing extra info in subwindows), the opaqueness ("alpha") of the window, whether the window is shown with borders or not, and whether or not the window will be shown on top of the other windows.

Below the entries for the contained terminal windows, is an entry, "Fullscreen" for toggling fullscreen mode for that application window. That entry will be gray when fullscreen mode is off and white when it is on.

In the primary application window which contains the main window, there is an entry, "Send Keypad Modifier", after that for whether key strokes from the numeric keypad will be sent to the game with the keypad modifier set. That entry will be gray when the modifier is not send and will be white when the modifier is sent. Sending the modifier allows some predefined keymaps to work, for instance shift with 8 from the numeric keypad to run north, at the cost of compatibility issues with some keyboard layouts that differ from the standard English keyboard layout for which normal keys have equivalents on the numeric keypad. https://github.com/angband/angband/issues/4522 has an example of the problems that can be avoided by not sending the keypad modifier.

Below "Send Keypad Modifier" in the primary application window's "Menu" is "Windows", use that to bring up one of the additional application windows.

The final two entries in "Menu" are "About" for displaying an information dialog about the game and "Quit" to save the game and exit.

When you leave the game, the current settings for the SDL interface

are saved as `sdl2init.txt` in the same directory as is used for preference files, see User Pref Files for details. Those settings will be automatically reloaded the next time you start the SDL interface.

Mac

With the Mac-specific front end, you can use Apple's standard mechanisms to control window placement: click and drag on a window's title bar to move it, click and drag on a window's edge or corner to change the window's dimensions, and click the red button at the top left corner of a subwindow to close it. To reopen a subwindow that you closed, use the Window menu from the Mac's menu bar while the game is the active application and select the entry near the bottom of that menu that corresponds to the subwindow you want to see. For a subwindow's entry to be enabled in the Window menu, that subwindow must be configured to display at least one category of information: see Showing extra info in subwindows for details.

To change the font for a window, click on the window's title bar and select "Edit Font" from the Settings menu in the Mac's menu bar. That will open a dialog which displays the family, typeface and size for the current font. Changing the selection for any of those will change the font in the window.

Whether the game's map is displayed as text or as graphical tiles can be set by selecting Settings from the Mac's menu bar while the game is the active application and then choosing from one of the entries in the Graphics option. Choosing "Classic ASCII" will display the map as text. Any of the other options will use some form of graphical tiles to display the map. If you wish to adjust how graphical tiles are scaled to match up with the currently selected font in the main window, select 'Change Tile Set Scaling…' in the Settings menu.

When you leave the game, the current Mac-specific settings are saved and will be automatically reloaded when you restart. The settings are stored in `Library/Preferences/org.rephial.angband.plist` within your user directory. If you suspect those settings have been corrupted in some way or would

like to start again from the default settings, quit the game if it is running, open a Terminal window (i.e. select 'Go->Utilities->Terminal' from the Finder's menus), and, in that Terminal window, run this:

```
defaults delete org.rephial.angband
```

to clear the contents of the preferences file and any cached preferences that may be retained in memory.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Version Information

Angband has been maintained and developed by a succession of volunteers since it was written in the early 1990s. The current maintainer is Nick McConnell.

The current version is 4.2.4. Detailed information about this version and previous versions can be found at http://angband.github.io/angband. Also additional information can be found at the angband forums (http://angband.oook.cz).

This file was last updated for Angband 3.1.2 and remains here mainly for historical purposes.

Angband has an incredibly complex history, and is the result of a lot of work by a lot of people, all of whom have contributed their time and energy for free, being rewarded only by the pleasure of keeping alive one of the best freeware games available anywhere.

The version control files, if they existed, would span more than ten years time, and more than six different primary developers. Without such files, we must rely on simpler methods, such as change logs, source file diffs, and word of mouth. Some of this information is summarised in this file.

Please be sure to read the ‘Copying and licence information’.

Brief Version History

First came “VMS Moria”, by Robert Alan Koeneke (1985).

Then came “Umoria” (Unix Moria), by James E. Wilson (1989).

Details about the history of the various flavors of “Moria”, the direct ancestor to Angband, can be found elsewhere, and a note from Robert Alan Koeneke is included in this file. Note that “Moria” has been ported to a variety of platforms, and has its own newsgroup, and its own fans.

In 1990, Alex Cutler and Andy Astrand, with the help of other students at the University of Warwick, created Angband 1.0, based on the existing code for Umoria 5.2.1. They wanted to expand the game, keeping or even strengthening the grounding in Tolkien lore, while adding more monsters and items, including unique monsters and artifact items, plus activation, pseudo-sensing, level feelings, and special dungeon rooms.

Over time, Sean Marsh, Geoff Hill, Charles Teague, and others, worked on the source, releasing a copy known as “Angband 2.4.frog_knows” at some point, which ran only on Unix systems, but which was ported by various people to various other systems. One of the most significant ports was the “PC Angband 1.4” port, for old DOS machines, which added color and various other significant changes, only some of which ever made it back into the official source.

Then Charles Swiger (cs4w+@andrew.cmu.edu) took over, sometime in late 1993, cleaning up the code, fixing a lot of bugs, and bringing together various patches from various people, resulting in several versions of Angband, starting with Angband 2.5.1 (?), and leading up to the release of Angband 2.6.1 (and Angband 2.6.2) in late 1994. Some of the changes during this period were based on suggestions from the “net”, and from various related games, including “UMoria 5.5”, “PC Angband 1.4”, and “FAngband”.

Angband 2.6.1 was primarily targeted towards Unix/NeXT machines, and it required the use of the low level “curses” commands for all screen manipulation and keypress interaction. Each release had to be ported from scratch to any new platforms, normally by creating visual display code that acted as a “curses” emulator. One such port was “Macintosh Angband 2.6.1”, by Keith

Randall, which added support for color, and which formed the basis for the first release of Angband 2.7.0.

During the last half of 1994, Ben Harrison had been playing with the Angband source, primarily to investigate the possibility of making some kind of automatic player for Angband, like the old "rogue-o-matic" program for the game "Rogue". The difficulty of compiling a version for the Macintosh, and the complexity of the code, prevented this, and so Ben began cleaning up the code in various ways for his personal use.

In late 1994, Charles Swiger announced that he was starting a real job and would no longer be able to be the Angband maintainer. This induced some amount of uproar in the Angband community (as represented by the Angband newsgroup), with various people attempting to form "committees" to take over the maintenance of Angband. Since committees have never given us anything but trouble (think "COBOL"), there was very little resistance when, on the first day of 1995, Ben made his code available, calling it "Angband 2.7.0", and by default, taking over as the new maintainer of Angband.

Angband 2.7.0 was a very clean (but very buggy) rewrite that, among other things, allowed extremely simple porting to multiple platforms, starting with Unix and Macintosh, and by the time most of the bugs were cleaned up, in Angband 2.7.2, including X11, and various IBM machines. Angband 2.7.4 was released to the "ftp.cis.ksu.edu" site, and quickly gained acceptance, perhaps helped by the OS2 and Windows and Amiga and Linux ports. Angband 2.7.5 and 2.7.6 added important capabilities such as macros and user pref files, and continued to clean up the source. Angband 2.7.8 was released to the major ftp archives as the first "stable" version in a year or so, with new "help files" and "spoiler files" for the "online help", plus a variety of minor tweaks and some new features.

After Angband 2.7.8 was released, Ben created a web site to keep track of all the changes made in each version (though a few may have been missed), and acquired the use of a new development ftp server to supplement the official "mirror" server. This web site is now permanently located at the Official Angband Home Page

(http://www.thangorodrim.net/). Unfortunately, the next six versions were numbered Angband 2.7.9v1 to Angband 2.7.9v6, but really each were rather major updates. Angband 2.8.0 and 2.8.1 were released using a more normal version scheme. Angband 2.8.2 and 2.8.3 add a few random features, clean up some code, and provide graphics support and such for a few more platforms.

After the release of Angband 2.8.3 Ben's free time was more and more occupied by his work. He released a beta version of Angband 2.8.5, introducing many new features, but couldn't give as much attention to maintaining the game as he wanted to. Meanwhile, an "unofficial" version by Robert Ruehlmann, incorporating three popular patches (the "Easy Patch" by Tim Baker, for opening doors and disarming traps without specifying the direction: Greg Wooledge's "Random Artifacts" patch: and Keldon Jones's "Optional Monster AI Improvement"), named "2.8.3h", was gaining popularity.

So in March 2000, Robert Ruehlmann offered to take over Angband and started to fix the remaining bugs in the Angband 2.8.5 beta. The resulting version was to be released as Angband 2.9.0. Further bugfixes and a couple of new features - including many in the realms of user-customizability, with greater control over ego-items, player races and classes, monsters, items and artifacts - have led to the current version.

And with the greater amount of user-customizability that is now possible, it was inevitable that SOMEBODY would eventually go and actually do something with it. Jonathan Ellis started customizing the user-editable text files in the 'edit' directory for his own personal use - originally, only by fixing bugs and inconsistencies (less powerful monsters being worth more experience per kill than more powerful ones, dragons doing a decent amount of damage in melee, monsters with two claws and one mouth getting one claw and three bite attacks, and so on).

At first, this was all that could really be done with it: adding new monsters and items was impossible, as the limits were fixed. And so only three new monsters made an appearance, each of them replacing an existing monster in the order: and one new artifact - "The Palantir of Westernesse". Gameplay balance could be tweaked

somewhat, by changing the level, power and rarity of certain items and monsters: and some changes were made, mostly with the attempt to reduce the notorious “triple whammy” effect of needing poison, confusion and nether resistance (or over 550 hps, if without nether resistance) all at once, straight after passing 2000’, forcing excessive scumming before this depth or risking unavoidable instant death: and then having nothing left to do but dive straight to 4000’ and scum for speed items, missing out on some of the most interesting depths of the dungeon. This problem, at least, could be addressed, but actual new things were less easy to add…

That all changed with Angband 2.9.1, which for the first time moved the limits themselves to a separate user-editable file, and allowed more monsters and items to be created without removing the old ones. At the same time, a patch by Matthias Kurzke was incorporated which allowed the creation of new ego-items. Various new powers, for the player and monsters, were added to the game - but no items or monsters yet had these powers (resist fear, poison brand, lose charisma, summon greater demons, and so on): indeed, arguably it could be said that the game had not even adjusted properly to Ben Harrison’s fractional speed system (Angband 2.7.0) or the addition of the other attack forms such as shards, sound, chaos, nexus and so on (even before Ben.)

The Official Angband Home Page (http://rephial.org/) serves not only as the most up to date description of Angband, but also lists changes made between versions, and changes planned for upcoming versions, and lists various email addresses and web sites related to Angband.

Some of the changes between Angband 2.6.1 and 3.0.6

It is very hard to pin down, along the way from 2.6.2 to 3.0.6, exactly what changes were made, and exactly when they were made. Most releases involved so many changes from the previous release as to make “diff files” not very useful, since often the diff files are as long as the code itself. Most of the changes, with the notable exception of the creation of some of the new ‘main-xxx.c’

files for the various new platforms, and a few other minor exceptions generally noted directly in comments in the source, were written by Ben or Robert, either spontaneously, or, more commonly, as the result of a suggestion or comment by an Angband player.

The most important modification was a massive “code level cleanup” for 2.7.x, largely completed in 2.7.8, that made all other modifications much simpler and safer. This cleanup was so massive that in many places the code is no longer recognizable, for example, via “diff -r”, often because it was rewritten from scratch.

The second most important modification was the design of a generic ‘z-term.c’ package, which allows Angband to be ported to a new machine with as few as 50 lines of code. Angband 2.9.3 thus runs without modification on many machines, including Macintosh, PowerMac, Unix/X11, Unix/Curses, Amiga, Windows, OS2-386, DOS-386, and even DOS-286.

It would be difficult to list all of the changes between Angband 2.6.1 and 3.0.6, because many of them were made in passing during the massive code level cleanup. Many of the changes are invisible to the user, but still provide increased simplicity and efficiency, and decreased code size, or make other more visable changes possible. For example, the new ‘project()’ code that handles all bolts, beams, and balls, the new ‘update_view()’ code that simplifies line of sight computation, or the new ‘generate()’ code that builds new levels in the dungeon. Many changes have been made to increase efficiency, including the new ‘process_monsters()’ and ‘update_monsters()’ functions, and the new ‘objdes()’ and ‘light_spot()’ routines. The generic ‘z-term.c’ package yielded efficient screen updates, and enabled the efficient use of color.

The most visible (to ordinary players) changes that happened as a result of Ben Harrison’s maintainership were:

- a far greater degree of user-customizability as shown by the ‘info.txt’ files
- the “fractional” speed system, with +10 in the new scheme equalling +1 in old money
- object stacking, the ability to have more than one object in a

square: first tried in 2.7.9, completed in 2.8.2.

It should also be pointed out at this point that the far cleaner nature of Ben's code as compared to previous versions has given many other people the opportunity to base code for their own Angband variants on it. And so a plethora of new variants have appeared, many of them far more different from Angband now than Angband ever was from Moria, and yet still based on Ben's coding ideals for Angband.

For Angband 2.9.0, the first few new visible features were a random artifact generator (originally developed from a variant by Greg Wooledge), an option to improve monster AI (believed to have originally started out life in a patch written by Keldon Jones), and a patch to allow easier handling of opening and closing doors and disarming traps (by Tim Baker). For Angband 2.9.1 has also come such things as the ability to increase the size of the editable text files and thus the number of monsters, artifacts, items, ego-items and vaults in the game (many new vaults were written by Chris Weisiger, some by others, and the number of vaults in the game at this time was doubled), and much greater customizability of ego-items has become possible thanks to a patch written by Matthias Kurzke. It is also now possible to add new character races to the game, and to edit the shopkeepers with respect to their greed, tolerance of haggling and reactions to the character based on his race. Angband 2.9.2 adds support for poison branded weapons to the game. Angband 2.9.3 made the character class itself customizable to an extent.

A Posting from the Original Author

> From: koeneke@ionet.net (Robert Alan Koeneke)
> Newsgroups: rec.games.roguelike.angband,rec.games.roguelike.moria
> Subject: Early history of Moria
> Date: Wed, 21 Feb 1996 04:20:51 GMT
>
> I had some email show up asking about the origin of Moria, and its relation to Rogue. So I thought I would just post some text on the early days of

Moria.

First of all, yes, I really am the Robert Koeneke who wrote the first Moria. I had a lot of mail accussing me of pulling their leg and such. I just recently connected to Internet (yes, I work for a company in the dark ages where Internet is concerned) and was real surprised to find Moria in the news groups… Angband was an even bigger surprise, since I have never seen it. I probably spoke to its originator though… I have given permission to lots of people through the years to enhance, modify, or whatever as long as they freely distributed the results. I have always been a proponent of sharing games, not selling them.

Anyway…

Around 1980 or 81 I was enrolled in engineering courses at the University of Oklahoma. The engineering lab ran on a PDP 1170 under an early version of UNIX. I was always good at computers, so it was natural for me to get to know the system administrators. They invited me one night to stay and play some games, an early startrek game, The Colossal Cave Adventure (later just 'Adventure'), and late one night, a new dungeon game called 'Rogue'.

So yes, I was exposed to Rogue before Moria was even a gleam in my eye. In fact, Rogue was directly responsible for millions of hours of play time wasted on Moria and its descendents…

Soon after playing Rogue (and man, was I HOOKED), I got a job in a different department as a student assistant in computers. I worked on one of the early VAX 11/780's running VMS, and no games were available for it at that time. The engineering lab got a real geek of an administrator who thought the only purpose of a computer was

WORK! Imagine… Soooo, no more games, and no more rogue!

This was intolerable! So I decided to write my own rogue game, Moria Beta 1.0. I had three languages available on my VMS system. Fortran IV, PASCAL V1.?, and BASIC. Since most of the game was string manipulation, I wrote the first attempt at Moria in VMS BASIC, and it looked a LOT like Rogue, at least what I could remember of it. Then I began getting ideas of how to improve it, how it should work differently, and I pretty much didn't touch it for about a year.

Around 1983, two things happened that caused Moria to be born in its recognizable form. I was engaged to be married, and the only cure for THAT is to work so hard you can't think about it; and I was enrolled for fall to take an operating systems class in PASCAL.

So, I investigated the new version of VMS PASCAL and found out it had a new feature. Variable length strings! Wow…

That summer I finished Moria 1.0 in VMS PASCAL. I learned more about data structures, optimization, and just plain programming that summer then in all of my years in school. I soon drew a crowd of devoted Moria players… All at OU.

I asked Jimmey Todd, a good friend of mine, to write a better character generator for the game, and so the skills and history were born. Jimmey helped out on many of the functions in the game as well. This would have been about Moria 2.0

In the following two years, I listened a lot to my players and kept making enhancements to the game to fix problems, to challenge them, and to keep them going. If anyone managed to win, I

immediately found out how, and ‘enhanced’ the game to make it harder. I once vowed it was ‘unbeatable’, and a week later a friend of mine beat it! His character, ‘Iggy’, was placed into the game as ‘The Evil Iggy’, and immortalized… And of course, I went in and plugged up the trick he used to win…

Around 1985 I started sending out source to other universities. Just before a OU / Texas football clash, I was asked to send a copy to the Univeristy of Texas… I couldn’t resist… I modified it so that the begger on the town level was ‘An OU football fan’ and they moved at maximum rate. They also multiplied at maximum rate… So the first step you took and woke one up, it crossed the floor increasing to hundreds of them and pounded you into oblivion… I soon received a call and provided instructions on how to ‘de-enhance’ the game!

Around 1986 - 87 I released Moria 4.7, my last official release. I was working on a Moria 5.0 when I left OU to go to work for American Airlines (and yes, I still work there). Moria 5.0 was a complete rewrite, and contained many neat enhancements, features, you name it. It had water, streams, lakes, pools, with water monsters. It had ‘mysterious orbs’ which could be carried like torches for light but also gave off magical aura’s (like protection from fire, or aggravate monster…). It had new weapons and treasures… I left it with the student assistants at OU to be finished, but I guess it soon died on the vine. As far as I know, that source was lost…

I gave permission to anyone who asked to work on the game. Several people asked if they could convert it to C, and I said fine as long as a complete credit history was maintained, and that it could NEVER be sold, only given. So I guess one or more of them succeeded in their efforts to rewrite it in C.

I have since received thousands of letters from all over the world from players telling about their exploits, and from administrators cursing the day I was born… I received mail from behind the iron curtain (while it was still standing) talking about the game on VAX's (which supposedly couldn't be there due to export laws). I used to have a map with pins for every letter I received, but I gave up on that!

I am very happy to learn my creation keeps on going… I plan to download it and Angband and play them… Maybe something has been added that will surprise me! That would be nice… I never got to play Moria and be surprised…

Robert Alan Koeneke
koeneke@ionet.net

Previous Versions (outdated)

VMS Moria Version 4.8

Version 4.8	03/25/83
Version 4.7	05/01/84
Version 4.6	07/10/84
Version 4.5	11/20/84
Version 4.4	01/20/85

Modules:

VAK	Dungeon Generator
CHAR	Character Generator
MOR	Moria Module
MISC	Miscellaneous
TOWN	Town Level & Misc
HELP	Internal Help & Misc
SCORE	Source Release Version

Robert Alan Koeneke
Student/University of Oklahoma

Umoria Version 5.2 (formerly UNIX Moria)

Version 4.83	5/14/87
Version 4.85	10/26/87
Version 4.87	5/27/88
Version 5.0	11/2/89
Version 5.2	5/9/90

James E. Wilson, U.C. Berkeley
wilson@ernie.Berkeley.EDU ...!ucbvax!ucbernie!wilson

Other contributors:

D. G. Kneller	MSDOS Moria port
Christopher J. Stuart	recall, options, inventory, and running code
Curtis McCauley	Macintosh Moria port
Stephen A. Jacobs	Atari ST Moria port
William Setzer	object naming code
David J. Grabiner	numerous bug reports, and consistency checking
Dan Bernstein	UNIX hangup signal fix, many bug fixes
and many others...	

Early Angband credits

Version 2.0
Alex Cutler, Andy Astrand, Sean Marsh, Geoff Hill, Charles Teague.

Version 2.4
05/09/1993

Version 2.5
12/05/1993 Charles Swiger

Version 2.6
09/04/1994 Charles Swiger

Version 2.7
    01/01/1995 Ben Harrison

Version 2.8
    01/01/1997 Ben Harrison

Version 2.9
    04/10/2000 Robert Ruehlmann

Contributors (incomplete)

Peter Berger, “Prfnoff”, Arcum Dagsson, Ed Cogburn, Matthias Kurzke, Ben Harrison, Steven Fuerst, Julian Lighton, Andrew Hill, Werner Baer, Tom Morton, “Cyric the Mad”, Chris Kern, Tim Baker, Jurriaan Kalkman, Alexander Wilkins, Mauro Scarpa, John I’anson-Holton, “facade”, Dennis van Es, Kenneth A. Strom, Wei-Hwa Huang, Nikodemus, Timo Pietilä, Greg Wooledge, Keldon Jones, Shayne Steele, Dr. Andrew White, Musus Umbra, Jonathan Ellis

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Copying and licence information

Angband is free software; you can redistribute it and/or modify it under the terms of either the GNU General Public License, version 2 [http://www.gnu.org/licenses/gpl-2.0.html] as published by the Free Software Foundation, or the “Angband licence”:

> Copyright (c) 1997 Ben Harrison, James E. Wilson, Robert A. Koeneke
>
> This software may be copied and distributed for educational, research, and not for profit purposes provided that this copyright and statement are included in all such copies. Other copyrights may also apply.

With the following exceptions:

- The SDL runtime libraries (if provided with your copy of the game) are under the following licence:

  > The Simple DirectMedia Layer (SDL for short) is a cross-platform library designed to make it easy to write multi-media software, such as games and emulators.
  >
  > The Simple DirectMedia Layer library source code is available from: http://www.libsdl.org/

This library is distributed under the terms of the GNU LGPL license: http://www.gnu.org/copyleft/lesser.html

- Adam Bolt's (16x16) graphics may be redistributed and used for any purpose, with or without modification.
- David Gervais' (32x32) graphics may be redistributed, modified, and used only under the terms of the Creative Commons Attribution 3.0 [http://creativecommons.org/licenses/by/3.0/] licence.
- Shockbolt's (64x64) graphics are distributed under the following licence:

> The Shockbolt Angband 64x64/128x64 tileset is copyright (C) Raymond Gaustadnes 2012. It can be found in the file lib/tiles/shockbolt/64x64.png.
>
> Permission is granted to:
>
> - use the tileset with in-development and released versions of Angband
> - distribute and make copies of the tileset with in-development and released versions of Angband, as long as no fee is charged for it
> - incorporate tiles designed by the author for variants of Angband and use and distribute them with Angband under the terms above
>
> Permission is not granted to:
>
> - modify the tileset without the author's permission.
> - incorporate tiles designed for ToME that do not appear in the Angband tileset.
> - use or distribute the tileset with other

games or projects. If you want to use and distribute the tileset with other games or projects, you must obtain explicit permission from the author. Non-commercial games or projects may be granted permission to use them, and if so, use will be allowed as long as the game or project remains non-commercial. To use them in commercial games, a non-exclusive licence must be acquired from the author.

Currently some of the tiles in the 64x64.png tilesheet were resized from tiles made by David Gervais for the 32x32 set.

- The sounds are licenced under the Creative Commons Attribution-NonCommercial-Sharealike licence. They were created by Dubtrain <angband@dubtrain.com>. You can find them in Wave format at http://www.dubtrain.com/angband/.

- The font files are all by Leon Marrick and/or Sheldon Simms III and/or Nick McConnell, all of whom have agreed to their Angband work being released under the GPL.

It is considered good practice to retain this statement for derivatives, rather than (e.g.) redistributing Adam Bolt's tiles under the GPL, or choosing to make a variant which is only under one of the Angband or GPL licences. This allows easier sharing of changes between variants.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Credits

Version 4.2.x

Code contribution:

- Adriankhl
- Adrian Siekierka (asiekierka)
- Alex Mooney
- Andre Maroneze
- Andrew York
- Anna Sidwell (takkaria)
- ArmiesAndCastles (v-chirkov)
- bacchist
- backwardsEric
- Ben Semmler (molybdenum)
- Bardur Arantsson
- Bill Peterson
- Binrui Dong
- bron
- Cameron Ball
- Colin Woodbury
- Cuboideb (Diego Gonzalez)
- Dag Arneson (sanedragon)
- Daniel Burgener
- David Medley
- Derakon
- Diego Herrera
- Eastwind921
- Elly Fong-Jones (elly)

- fizzix (Aaron Bader)
- fruviad
- jdholbrook81
- jefetienne
- Joan Andrés
- John Weismiller (emar)
- Kusunose Toru
- Lars Haugseth
- magnate (Chris Carr)
- MarbleDice
- Michael Courtney (wobbly)
- Mikolaj Konarski
- Paul Johnson
- piels
- PowerWyrm
- pwinckles (moosferatu)
- Rodent/Sideways/sulkasormi
- Ryan Schmidt
- Shanoah Alkire
- spenserblack
- Stefan Strogin
- tangar
- Tim Schumacher (timschumi)
- Vic K
- wkmanire
- wobbly
- Yutao Yuan (infmagic2047)

Thanks are due to many people for, reporting bugs, suggesting changes and contributing to discussion which has influenced this version. The following is an (undoubtedly poor) attempt to list you all, please let me know if you think you ought to be included:

- Adam
- Anarchic Fox
- animal_waves
- AnonymousHero
- Antoine
- archolewa
- Aszazin

- Atriel
- bio_hazard
- Bogatyr
- bughunter
- bunnies
- Capn_Carpaccio
- Carg
- Carnivean
- cccfire
- Chud
- cjslates
- ClaytonAguiar
- Clearshade
- clouded
- Cold_Heart
- Combatereak
- Darin
- debo (cyberdemons pls)
- desstorm
- dionysian
- Djbanete
- dos350
- Dragget
- drquicksilver
- Ed_47569
- Egavactip
- emulord
- EpicMan
- Eric
- Estie
- Estragon
- Evilpotatoe
- ewert
- Flambard
- floatRand
- FogSpear
- fph
- Gauss
- Geoff Hill (yes, that one)
- geoff_tewierik

- gglibertine
- Glorfindel
- Goaticus
- Grotug
- gtrudeau88
- Gwarl
- half
- HallucinationMushroom
- HebrewToYou
- Holy_Rage
- Hounded
- Hrrunstar
- Huqhox
- ImEsteban
- Ingwe Ingweron
- JBright
- Jeff Greene (nppangband)
- jevansau
- jml34
- jsv
- Julian (jl8e)
- Jungle_Boy
- kandrc
- kaypy
- khearn
- kineahora
- Kinematics
- lanactoor
- lonadar
- luneya
- Mark
- MattB
- Mike
- misanthropope
- MITZE
- MKula
- Mondkalb
- Monkey Face
- Moving Pictures
- mrfy

- Muscleguy
- MWGE
- Narry
- NCountr
- Netbrian
- NightLizard
- Nomad
- olivertheorem
- Once
- Oraticus
- Pahasusi
- Patashu
- Pete Mack
- Philip
- Pondlife
- Pussy Galore
- quarague
- Quirk
- Raerick
- Raxmei
- renato
- robinjohnson
- Rydel
- Saru
- Scatha
- schatz
- scrarth
- Selkie
- shirish
- Sinquen
- Sky
- smbhax
- spara
- Sparrow the Dunadan
- Sphara
- swaggert
- the Invisible Stalker
- Therem Harth
- Thraalbee
- Tibarius

- Timo Pietilä
- TJA
- TJS
- topazg
- Torr
- Ugramoth
- Vivit
- Voovus
- Vorczar
- Werbaer
- whartung
- Whelk
- will_asher
- WindLord
- Wiwaxia
- Xaxyx
- Youssarian
- Zikke
- Zirael

Special thanks to Luke McConnell for many, many conversations on game design which influenced design decisions across the breadth and depth of Angband.

Previous maintainers

Angband 3.0.8 - 3.5.1:
Anna Sidwell <anna at takkaria.org>

Angband 2.9.0 - 3.0.6:
Robert Ruehlmann <rr9 at thangorodrim.net>

Angband 2.7.0 - 2.8.5:
Ben Harrison <benh at phial.com>

Angband 2.0 - 2.4 - 2.6.1:
Alex Cutler, Andy Astrand, Sean Marsh, Geoff Hill, Charles Teague, Charles Swiger

Contributors

Many thanks go to the following people who have contributed patches, bugfixes, and other stuff for Angband prior to 4.0:

Peter Berger, Andrew Hill, Werner Baer, Tom Morton, “Cyric the Mad”, Chris Kern, Jurriaan Kalkman, Alexander Wilkins, Mauro Scarpa, “facade”, Dennis van Es, Kenneth A. Strom, Wei-Hwa Huang, Nikodemus, Timo Pietilä, Shayne Steele, Dr. Andrew White, Greg Flint, Christopher Jeris, Ian Parkhouse, “Warhammer”, Scott Holder, Brent Ross, Kazuo Ito, Willem Siemelink, “Luthien”, David J. Grabiner, Ilya Bely, “chungkuo”, Kieron Dunbar, George W. Harris, Joseph Oberlander, Paul Moore, Andreas Tophinke, Leon Marrick, Peter J. Rowe, Wim Benthem, Jaroslav Sladek, Keith Perkins, Hugo Kornelis, Pete Mack, Marco K, Frank Palazzolo, Christer Nyfalt, Andrew Doull, Kenneth Boyd, Iain McFall, Christophe Cavalaria, Brendon Oliver, “Zaxx”, “theninja”, “Twilight Forest”, “jbu”, “AnonymousHero”, Stefan O’Rear, “SilverD”, Ed Graham, Tobias Franke, “rhinocesaurus”, “Bron”, “Mangojuice”, Chris Robertson, Joe Buck, “tigen”, “Big Al”, Paul Blay, J. D. White, Rowan Beentje, “pelpel”, Shanoah Alkire, Alexander Philips, “mikon”, “Antoine”, “Irashtar”, “roustk”, Diego Gonzalez, Takeshi Mogami, Julian Lighton, Aram Harrow, William Tanksley, Chris Ang, Dean Anderson, Daniel Nash, David Blackston, Heino Vander Sanden, Mark Kvale, Sheldon Simms, Topi Ylinen, “Gileba”, Jeff Greene, Joshua Middendorf, Tom Demuyt, Alexander Ulyanov, Alexander Malmberg, Chris R. Martin, Chris Herborth, Craig Oliver, “DarkGod”, David Boeren, David DeLaney, David Kahane, Dennis Payne, Desvignes Sebastien, Ekkehard Kraemer, Eugene Hung, HansJoachim Baader, Heiko Herold, John Rauser, Jonathan Sari, Joseph William Dixon, Joseph Hall, John M. Kewley, Ken Wigle, Keith H. Randall, Kevin Bracey, Mike Marcelais, Maarten Hazewinkel, Peter Ammon, Peter Seebach, Randy Hutson, Scott Egashira, Skirmantas Kligys, Steve Linberg, Silas Dunsmore, Tom Harris, Ron Anderson, Ross E. Becker, Denis Eropkin, Torbjorn Lindgren, Lars Haugseth, Jon Taylor, Roland Jay Roberts, “Sergey”, “cb”, Michael Pope, “hmj”, Colin Spry, Ed Cogburn, “Yendor”, Thomas Dedorson, “Ewert”, Rooslan S. Khayrov, Thapper, “Max

Stats", "SSK", "ChodTheWacko", "Zaxx", Jonas Lith, Jens Schou, "Lebannen", Daniel Santos, Edd Barrett (vext01), mtadd, Peter Denison (noz), Kiyoshi Aman (Aerdan), David Barr (david3x3x3), Chris Weisiger (Derakon), Buzzkill, Scott Michael, LastQuestion, danial.santos, LuthienCeleste, shadowsun

Raymond "Shockbolt" Gaustadnes
    The Shockbolt tiles.

Greg Wooledge <greg at wooledge.org>
    Basic autoconf support, the original random artifact generator, and various ideas for rebalancing the game including the new list of magic spells

Tim Baker <dbaker at direct.ca>
    Made the "easy patch" and organized the patches for the Angband 2.8.5 beta

Eytan Zweig <eytanzw at yahoo.com>
    Many bug reports and patches

Jonathan Ellis <jonathan at franz-liszt.freeserve.co.uk>
    Updated edit and help files; added tons of new monsters, artifacts, vaults, objects, as well as a new player race, and rebalanced many things

John I'anson-Holton <jianson at milbank.com>
    Many bugfixes and patches

Steven Fuerst <sfuerst at physics.usyd.edu.au>
    Improved X11, XAW, and GTK code

"Bablos" <angband at blueyonder.co.uk>
    Updated Amiga code

Matthias Kurzke <mawende at gmx.net>
    Ego-item patch and various code changes for the JLE patch

Keldon Jones <keldon at umr.edu>
    Improved Monster AI

Adam Bolt

16x16 tiles

Arcum Dagsson
: Configurable artifact activations

“Prfnoff”
: Customizable player races, player history, shop owners, …

Mark Howson
: Improvements to the Amiga code

Musus Umbra
: Improvements to the Acorn RISC OS code

Hallvard B. Furuseth
: Many improvements to the autoconf support, code-cleanups, and tons of bugfixes

Kusunose Toru
: Various bugfixes

Eddie Grove
: Bugfixes, patches and radical ideas too numerous to count (but in particular for ID-by-use).

Nomad
: 8x16 tiles, loads of new room templates

The UPX team (Markus Oberhumer and Laszlo Molnar)
: The UPX packer for executables http://upx.tsx.org/ is used to reduce the size of the Windows and DOS binaries.

qwerty
: LaTeX-based help file generation

Federico Poloni (fph)
: Manual and documentation updates, formatting in reStructuredText

Peter Ammon (ridiculous_fish)
: Rewritten OSX main-cocoa interface

William Moore (MarbleDice)
: Bitflag code and numerous other improvements and fixes during 3.1.x

Antony Sidwell (ajps)
: Default point-based stat allocations, and numerous UI improvements, original core-UI split code

“PowerWyrm”
: Numerous bug fixes and code improvements

Version 4.0.x

Code contribution:

- Aaron Bader (fizzix)
- Antony Sidwell (ajps)
- Andi Sidwell (takkaria)
- Bardur Arantsson
- Ben Semmler (molybdenum)
- Chris Carr (magnate)
- Christian Heckendorf
- Elly Fong-Jones (elly)
- Elsairon
- Erik Osheim (d_m)
- flaviommedeiros
- Jagath Samarabandu
- Jose Antonio Dura
- Kevin J. Fletcher
- LostTemplar
- Michel Carroll
- Nick McConnell
- Nomad
- Peter Denison (noz)
- phantom-voltage
- PowerWyrm
- redlumf
- Robert Au (myshkin)
- Rydelfox
- Timothy Collett

Beta testing and bug reporting/fixing:

- Ingwe Ingweron
- Nomad
- MattB
- Thraalbeast
- tumbleweed
- AndyHK
- Rhonwyn
- Jungle_Boy
- Darin
- StMicah
- debo
- pen
- topazg
- wobbly
- DeusIrae
- Timo Pietilä
- ranger jeff
- passer_by
- Runaway1956
- mrrstark
- Estie
- shreesh
- elliptic
- Gorbad
- letslaugh
- ShadowTechnology
- bryan.g.hutchinson
- Werbaer
- fph
- yyt16384
- kandrc
- Nivra
- Tarrasque
- Egavactip
- zog
- troycheek

Version 4.1.x

Code contribution:

- Alex Mooney
- Andi Sidwell (takkaria)
- AndreyB
- Bardur Arantsson
- Ben Semmler
- crayonsmelting
- Derakon
- Erik Osheim (d_m)
- fizzix (Aaron Bader)
- Flavio Medeiros
- Graeme Russ
- Gwilim Owen
- Jean-François Caron
- kaypy
- Kevin J. Fletcher
- Nomad
- Pete McIlroy
- Peter (Hermann Döppes)
- Peter McIlroy
- phantom-voltage
- PowerWyrm
- rmzelle
- rowanbeentje
- Tiara Smith
- Twisted Pair in my Hair
- Vic K (t4nk)
- William Orr

Thanks are due for contributing to discussion which has influenced this version to a great many people (too many to list) on

- the Angband forums (http://angband.oook.cz/forum/)
- the #angband-dev and #angband IRC channels on libera.chat
- the roguelikes subreddit (https://www.reddit.com/r/roguelikes/)
- the #band channel on the roguelikes discord

Special thanks to Luke McConnell for many, many conversations on game design which influenced design decisions across the breadth

and depth of Angband.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Modifying Angband

Angband is not just a great game in its own right, it is really easy to modify. Much of the detail of the game is contained in text data files. These can be changed using nothing more than a text editor for an immediate change to how the game works.

These data files are in lib/gamedata.

Each file has a header which describes the lines which make up entries of the file, and for the most part this will make it clear what needs to be done to make changes to the files. Below is brief description of each of the files.

Those who want to change the game more than is allowed just by varying the data files will need the source code. Below the data file descriptions is a brief discussion of where to start on such an endeavour.

The data files

constants.txt
: This file contains game values such as carried item capacity, visual range and dungeon level and town dimensions.

object_base.txt
: This file contains the names and common properties of the basic object classes - scroll, sword, ring, and so on. All objects have an object base. Each object base is assigned a ‘tval’ - a numeric index. The tvals are defined in src/list-tvals.h. While adding new object bases is possible, it is unlikely to do much

without deeper changes to the game.

object.txt
: This file contains the names, properties and description of all the object types that appear in Angband. New object kinds can easily be added to this file, or existing ones edited. Each object defined by this file has an object base, and is also allocated another numeric index called an 'sval'. A tval-sval pair completely identifies an object - since the tval and sval are saved to savefiles, removing or adding objects is likely to render existing save files unusable.

ego_item.txt
: This file contains the names, properties and description of ego items, which are magically enhanced weapons and armour. New ego items can be added or removed at will, although removing or changing one with an instance currently in the game might cause problems.

artifact.txt
: This file contains the names, properties and description of artifacts, which are unique items - only one of each will ever be generated. If you are considering major changes, new artifacts are one of the most visible signs of a change of theme. Regardless, new artifacts are easy and fun to design.

names.txt
: This file contains lists of words which are used to generate names for random character names, random artifacts and scrolls. Again, in the case of a change of theme, this is a good way of displaying the new theme.

activation.txt
: Activations are used for artifacts and some regular objects, and could be used for ego items (although none currently are). Some standard artifacts from artifact.txt have activations, and random artifacts may have any activation from this file chosen for them. Activations can be made up of any effects (see list-effects.h and effect-handler-*something*.c).

flavor.txt

Items such as potions and wands are assigned a flavor per object kind, different in each game. There need to be at least as many flavors for each flavored object base as objects with that base.

monster_base.txt
: Monster bases are the monster equivalent of object bases - classes of monster like orc, troll or vampire. This file contains the properties common across all monsters in each of these classes.

monster.txt
: This contains the detail of all monster races, each of which will have its monster base properties plus additional ones. Some monsters are unique, and once killed will never reappear.

monster_spell.txt
: All the spells that can be cast by monsters (and are referred to in the 'spells:' lines in monster.txt) are defined in this file. As with activations, monster spells are built up from effects.

pain.txt
: This file contains the various messages that are given to describe how a monster responds to attack.

pit.txt
: Dungeon levels can contain pits - rooms full of a particular selection of monsters. This file defines these selections. They can also be used, for example, to generate partial or complete dungeon levels with themed monsters.

class.txt
: This file completely defines how player classes work, including all details of castable spells. There are some class-specific properties hard-coded, which are referred to via the 'flags:' lines, and appear in list-player-flags.h.

p_race.txt
: This file defines all player race characteristics. Race-specific code is handled as for classes.

body.txt
: Every player race has a body, which defines what equipment they can use. Currently there is only one body, which all races use, but this is easily changeable for significant effect.

history.txt
: This file is for creating the player background found on the character screen. If a new race is introduced, a selection of background information for it will need to be added.

hints.txt
: This is simply a list of general pieces of advice that shopkeepers will give to their customers.

quest.txt
: This file defines the quest monsters (Sauron and Morgoth) and where they appear. This currently can't easily be changed, as there are still hard-coded aspects of the quests.

terrain.txt
: This file defines the kind of terrain which can appear in Angband, and its properties. Current terrain can be changed (with possibly large effects), but removing it without code changes is likely to break the game. Adding new terrain will have no effect by itself, because there is no mechanism for it to appear.

trap.txt
: This defines all traps, door locks and runes. Actual trap effects appear in list-effects.h and effect-handler-*something*.c.

room_template.txt
: This is a list of templates for interesting-shaped rooms which appear in the dungeon. These can easily be changed and new ones added.

vault.txt
: Similar to room_template.txt, this handles vaults, which are very dangerous and lucrative rooms.

dungeon_profile.txt

This file contains fairly technical details about the different types of dungeon level which can be generated. The actual generation routines are in gen-cave.c; the information here consists of parameters for generating individual levels, and for how often given level types appear.

store.txt
: This details the shop owners and their relative generosity.

blow_effects.txt
: This defines effects to the player caused by monster attacks. The simplest monster attacks just deal damage, but others can affect the player's status, stats or inventory.

blow_methods.txt
: This details the different ways monsters can attack (hit, claw, etc.). It affects the messages the player gets, and also whether the blow can stun or cut the player.

brand.txt
: This details how weapon brands work.

slay.txt
: This details how weapons can be more effective against certain monsters.

curse.txt
: This file contains all the different curses that can be applied to objects. It includes what type of object they can be applied to, random effects they can cause, and how they change an object's properties.

object_property.txt
: This file gives details about what properties an object can have (apart from basic combat and armor class). Every property has a code which is used in the game to refer to that property in some way. This means it is not possible to add new properties to this file and expect to have any effect, but it is possible to change how existing properties work.

player_timed.txt

This file defines some of the properties of timed effects (such as haste and confusion) that can apply to the player. It chiefly contains the messages on changes in these effects, and player attributes which prevent the effects. To add new timed effects or change the way existing ones operate, you will have to alter src/list-player-timed.h and probably other files, and re-compile the game.

projection.txt
: This file contains a lot of the defining information about projections - effects which can be produced at a distance by player or monsters, and affecting player, monsters, objects, and/or terrain. In particular, this file defines details of the effects of elemental attacks (such as fire or shards) and the effectiveness of corresponding player resistance. New projections have to be either included in src/list-elements.h (for elemental attacks) or included in src/list-projections.h (for all other projections), and the code to implement their effects put in other source files - src/project-obj.c for effects on objects, and other similarly-named files.

realm.txt
: This contains a small amount of information about the two current magic realms.

summon.txt
: This contains definitions for the types of monsters that can be summoned. Adding a new summon type is not yet possible, because the summon spells are hard-coded in src/list-mon-spells.h.

ui_entry.txt
: Defines entries that will be displayed in the second part of the character sheet and in the knowledge menu's equipable comparison. You can modify properties in object_property.txt and project_property.txt to bind them to those entries. The intent is to make it possible to add or remove a property without having to update ui-player.c or ui-equipcmp.c in addition to the changes necessary to have that property affect core gameplay.

ui_entry_base.txt
: Provides templates for use by ui_entry.txt.

ui_entry_renderer.txt
: Defines techniques, referenced in ui_entry.txt, for rendering a property in the character sheet or equipable comparison. While it is possible to add something that simply uses different palettes of symbols or colors than one of the current renderers, the basic rendering techniques are hard-coded in list-ui-entry-renderers.h.

ui_knowledge.txt
: Handles some configuration of the knowledge menus, namely the layout of the monster categories.

Making Graphical Tilesets

You can make new graphical tilesets for Angband or customize existing ones. In this section we'll dive into how tilesets are defined and describe how to set one up from scratch. First, we'll enumerate the steps required and then we'll break down each step in detail.

1. Create a directory to contain the tileset's files: (ex. `lib/tiles/mytileset`)
2. Register the tileset in `lib/tiles/list.txt`
3. Create an empty bitmap image large enough to hold your tileset
4. Store the empty bitmap image in your tileset folder
5. Author one or more `.prf` files to inform Angband how to use your tileset
6. Create a Makefile in your tileset folder

First you need to create a directory to contain your tileset's files. Put the directory in lib/tiles and choose a name for the directory that is lower-case and generally matches the naming convention of the other tilesets you see there. Once the directory has been created, the next step is to decide how big the tiles will be in pixels and then create a blank PNG image large enough to hold all of the tiles (be sure to enable alpha transparency). As an example, Shockbolt's tileset uses 64x64 pixel tiles. It also uses the special

alpha blending flag so it can use double-height tiles (64x128) for large or tall monsters. Its dimensions are 8192x2048 but the tileset is not completely full. More tiles can be added without increasing the size of the image as new objects are added to future releases of Angband. This should be kept in mind as packing your tileset into the smallest possible image size may not be the most maintainable solution. Be sure to name the image file after the tile size, for example 64x64.png. Use the base size even if you are enabling double-height tiles.

The only file you'll need to edit outside of your tileset's directory is lib/tiles/list.txt. list.txt contains a registry of which tilesets to load as well as some information about the size of the tiles and any special flags to set. The format of the file is documented in list.txt's header. Specifically, you will be defining the name of the tileset, which directory contains the tileset's files, how big the tiles are in pixels (i.e. 64x64), the name of the main preference file for the tileset and some additional flags which have to do with alpha blending. Not all tilesets need to set extra flags.

Now that the basic setup is complete you need to tell Angband how to interpret your tileset image. You need to map each tile in your image to a specific element in the game so that Angband knows which tiles to show for which ASCII characters. This process can be done incrementally because Angband will continue to show the default character symbols in-game for objects that have not yet been mapped. This is especially helpful for verifying that your tileset has been setup correctly before beginning to map things out in earnest. It also means that if new objects are added to the game that you have not mapped into your tileset, the game will still be playable with your tileset, albeit the displayed ASCII character may appear incongruous with your styling. Mapping tiles to game elements is done in text files called preference files which have the extension '.prf'.

The first thing to understand about mapping game elements in preference files is that everything that can be displayed in the game has a name, or in the case of flavors, an ID number. The names for each type of thing can be referenced from the data files as mentioned above. The table below is a quick reference for where to

find names of things and how to form IDs correctly to reference them.

Type	File	Example
Terrain	terrain.txt	feat:DOOR
Trap	trap.txt	trap:pit
Object	object.txt	object:light
Monster	monster.txt	monster:Kobold
Spell	spell_effect.txt	spell:fireball
Player	player_below.txt	player:<player>

Player pictures are referenced differently than other types of objects. They use a special query syntax that checks to see what kind class the player is as well as the gender in order to determine which picture to show. The query to select which tile to show for a female elf ranger would be:

```
?:[AND [EQU $CLASS Ranger] [EQU $RACE Elf]  [EQU $GENDER
```

Here, the query is checking to see if the player is a female Half-Elf and would use the assignment on the next line of the preference file only if this is true.

Some types of objects such as terrain can use different tiles based on their state. In the case of terrain, the terrain can have different images for when it is lit by a torch, or dark. these are selected by appending another colon and a specifier to the name. For example, this would be the name of a torch-lit up staircase:

```
feat:LESS:torch
```

It is possible to specify the same tile be used for all possible states of a terrain feature by using an asterisk. This example identifies any unknown terrain tile (a tile the player hasn't lit or otherwise seen yet):

```
feat:NONE:*
```

Given the full name of an object the last thing to do is to specify which tile from the tileset to use. Tile locations are given in a coordinate system using pairs of hexadecimal numbers. The

coordinates start from 0x80:0x80 and increment from there. The pairs translate directly to the top and left most pixel of the corresponding tile from the graphics file, so the top left pixel of the first tile on the top left of the graphics file would be specified as 0x80:0x80 (the pixel at x:0 y:0). The next tile immediately to the right of the that one would be 0x80:0x81. The tilesheet is sliced into rows and columns based on the tile size you specified in list.txt. So given a tile size of 64x64 pixels, the tile at 0x80:0x81 would be located in the graphics file at pixel x:64 y:0. Remember, the coordinates in the preference files are in hexadecimal, so the next number after 0x89 would be 0x8A. The next number after 0x8F would be 0x90 and so on. To map an object to your tileset you will add one complete line to the file per object. This example maps the tile at 0x81:0x81 to the terrain feature 'quartz vein' when the quartz vein is lit by torch light:

```
feat:QUARTZ:torch:0x81:0x81
```

Before going any further, it is advisable to map a single object in your preference file, then start the game up, select your tileset and make sure you see your mapped tile in game. If this worked, then you are ready to design and map the rest of your tiles. A quick example would be to map a tile for your home in the town to the first tile position in your graphics file:

```
feat:HOME:*:0x80:0x80
```

It's possible to have more than one preference file by using a sort of include syntax that causes other preference files referenced from your main preference file to also be read. It is also possible to place comments in your preference files to help you keep track of where different kinds of objects are mapped. Any text on a line after a `#` symbol is ignored. Shockbolt's tiles make great use of this and define a well organized set of mappings using three files with comments for each logical section of objects to be mapped:

```
# This is a comment
%:other-stuff.prf  # Load another preference file
```

The last step to take is to make sure your tileset will be packaged with Angband when it is compiled for distribution and that it can

be installed alongside the other tilesets. to do this you will need to add a file called ‘Makefile’ to your tileset directory. Copy and paste an existing Makefile from one of the other tileset directories and update the DATA and PACKAGE lines to match the filenames you chose for your tileset.

Once you have a working tileset and functional understanding of how tilesets are managed and organized, it would be a good idea to study Shockbolt’s tileset and follow the examples there in order to produce a high-quality tileset that you will be proud to share with others.

Larger changes

If changing data files is not enough for you, you will need to change actual game code and recompile it. The first place to look is in the compiled data files, some of which have already been mentioned:

list-compile-flags.h
list-effects.h
list-object-flags.h
list-object-modifiers.h
list-options.h
list-origins.h
list-player-flags.h
list-player-timed.h
list-projection-flags.h
list-monster-properties.h

Beyond this, you will have to have some knowledge of the C programming language, and can start making changes to the way the game runs or appears. Many people have done this - there are over 100 variants of Angband: http://angbandplus.github.io/AngbandPlus/ Should you get to this point, the best thing to do is to discuss your ideas on the Angband forums at http://angband.oook.cz. The people there are typically keen to hear new ideas and ways to play.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Compiling Instructions

The methods for compiling Angband vary by platform and by build system. If you get Angband working on a different platform or build system please let us know so we can add to this file. Unless otherwise noted, all the commands listed are to be run from top-level directory of the Angband source files.

Contents

- Cross-building for DOS with DJGPP
- Documentation

macOS

To build the new Cocoa front-end:

```
cd src
make -f Makefile.osx
```

That'll create a self-contained Mac application, Angband.app, in the directory above src. You may use that application where it is or move it to wherever is convenient for you.

By default, the current Makefile.osx builds an application that'll run natively on x86_64 or arm64 machines. If only one of those architectures is of interest to you or the version of Xcode you have doesn't support building both (a typical error message in that case is something like `/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/sys/cdefs.h:784:2: error: Unsupported architecture`), you can change the architectures built by setting ARCHS on the command line of make. To only build for x86_64, for instance, you would use:

```
cd src
make -f Makefile.osx clean
make -f Makefile.osx ARCHS=x86_64
```

(the clean step is to ensure that nothing from a previous build would cause trouble; you'll typically need to do that if you've built it before and then want to change the set of architectures to use). To build for multiple architectures, use a list of architectures separated by whitespace, which you'll have to quote. This is the equivalent to what Makefile.osx does by default:

```
make -f Makefile.osx ARCHS="x86_64 arm64"
```

Debug build

This will generate a debugging build like that described in the Linux

section:

```
cd src
make -f Makefile.osx clean
make -f Makefile.osx OPT="-g -O1 -fno-omit-frame-pointer
```

The clean step is there to clean out object files that were compiled with the default options. The “-g” adds debugging symbols. “-O1 -fno-omit-frame-pointer” dials back the optimization to get call stack traces that are easier to interpret. For even clearer call stack traces, you could add “-fno-optimize-sibling-calls” to the options or omit optimization entirely by replacing “-O1 -fno-omit-frame-pointer” with “-O0”. “-fsanitize=address -fsanitize=undefined” enables the AddressSanitizer and UndefinedBehaviorSanitizer tools.

To run the generated executable under Xcode’s command-line debugger, lldb, do this if you are already in the src directory from the compilation step:

```
cd ../Angband.app/Contents/MacOS
lldb ./angband
```

Test cases

To compile and run the unit tests, do this:

```
cd src
make -f Makefile.osx tests
```

If you want to rerun just one part, say monster/attack, of the unit tests, that’s most easily done by:

```
cd src/tests
monster/attack.exe
```

Previous versions put the test executables in src/tests/bin. With those versions, the second line above would be:

```
bin/monster/attack
```

The reason for changing directories to src/tests is to match up with

how the tests were compiled: they expect Angband's configuration data to be in ../../lib.

Statistics build

The Mac front end bypasses main.c and can not use the statistics front end. It is possible to enable the debugging commands related to statistics (see the descriptions for `S`, `D`, and `P` in Dungeon). To do so, include "-DUSE_STATS" in the setting for OPT passed to make. The equivalent of the standard build with those debugging commands enabled would be:

```
cd src
make -f Makefile.osx OPT="-DUSE_STATS -O2"
```

If you had already built everything without statistcs enabled, you would need to run either "rm wiz-stats.o" or "make -f Makefile.osx clean" immediately after running "cd src".

Linux / other UNIX

Native builds

Linux builds using autotools. There are several different front ends that you can optionally build (GCU, SDL, SDL2, and X11) using arguments to configure such as –enable-sdl, –disable-x11, etc. Each front end has different dependencies (e.g. ncurses, SDL libraries, etc).

If your source files are from cloning the git repository, you'll first need to run this to create the configure script:

```
./autogen.sh
```

That is not necessary if your source files are from the source archive, a .tar.gz file, for a release.

To build Angband to be run in-place, then run this:

```
./configure --with-no-install [other options as needed]
```

```
make
```

That'll create an executable in the src directory. You can run it from the same directory where you ran make with:

```
src/angband
```

To see what command line options are accepted, use:

```
src/angband -?
```

Note that some of Angband's makefiles (src/Makefile and src/tests/Makefile are the primary offenders) assume features present in GNU make. If the default make on your system is not GNU make, you'll likely have to replace instances of make in the quoted commands with whatever will run GNU make. On OpenBSD, for instance, that is gmake (which can be installed by running "pkg_add gmake").

On systems where there's several C compilers, ./configure may choose the wrong one. One example of that is on OpenBSD 6.9 when building Angband with SDL2: ./configure chooses gcc but the installed version of gcc can't handle the SDL2 header files that are installed via pkg_add. To override ./configure's default selection of the compiler, use:

```
env CC=the_good_compiler ./configure [the appropriate co
```

Replace the_good_compiler in that command with the command for running the compiler that you want. For OpenBSD 6.9 when compiling with SDL2, you'd replace the_good_compiler with cc or clang.

To build Angband to be installed in some other location, run this:

```
./configure --prefix /path/to [other options as needed]
make
make install
```

On some BSDs, you may need to copy install-sh into lib/ and various subdirectories of lib/ in order to install correctly.

Compilation with CMake

The compilation process with CMake requires a version greater than 3, by default the compilation process uses the X11 front end unless one or more of the other graphical front ends are selected. The graphical front ends are: GCU, SDL, SDL2 and X11. All of the following generate a self-contained directory, build, that you can move elsewhere or rename. To run the result, change directories to build (or whatever you renamed it to) and run ./Angband .

To build Angband with the X11 front end:

```
mkdir build && cd build
cmake ..
make
```

If you want to build the X11 front end while building one of the other graphical front ends, the option to pass to cmake is -DSUPPORT_X11_FRONTEND=ON .

To build Angband with the SDL front end:

```
mkdir build && cd build
cmake -DSUPPORT_SDL_FRONTEND=ON ..
make
```

To build Angband with the SDL2 front end:

```
mkdir build && cd build
cmake -DSUPPORT_SDL2_FRONTEND=ON ..
make
```

To build Angband with the GCU front end:

```
mkdir build && cd build
cmake -DSUPPORT_GCU_FRONTEND=ON ..
make
```

On OpenBSD (at least with OpenBSD 6.9), there's known issues with detecting the software needed for the GCU front end. As a workaround, you could use this instead:

```
mkdir build && cd build
mkdir -p ncursesw/include/ncursesw
ln -s /usr/include/ncurses.h ncursesw/include/ncursesw
mkdir -p ncursesw/lib
ln -s /usr/lib/libncursesw.so* ncursesw/lib
cmake -DCMAKE_PREFIX_PATH=`pwd`/ncursesw -DSUPPORT_GCU_F
make
```

You can build support for more than one of the graphical front ends by setting all the desired SUPPORT_*_FRONTEND options when running cmake (the exception to this are the SDL and SDL2 which can not be built at the same time). If you want the executable to have support for sound, pass -DSUPPORT_SDL_SOUND=ON or -DSUPPORT_SDL2_SOUND=ON to cmake (as with the SDL and SDL2 front ends, you can't build support for both SDL and SDL2 sound; it is also not possible to build the SDL front end with SDL2 sound or the SDL2 front end with SDL sound).

There are options to not build a self-contained installation and, instead, organize the files for a typical Linux or Unix layout. One such option installs the executable as setgid so the high score and save files can be stored in a centralized location for multiple users. To enable that option, pass -DSHARED_INSTALL=ON to cmake. To specify the group used for the setgid executable, pass -DINSTALL_GROUP_ID=xxx to cmake where you replace xxx with the name or number of the group to use. If you do not set the group, the games group will be used. Another option creates a read-only installation with any variable state, including the high score and save files, stored on a per-user basis in the user's own directories. To enable that option, pass -DREADONLY_INSTALL=ON to cmake. Turning on both SHARED_INSTALL and READONLY_INSTALL is not supported and will cause cmake to exit with an error. Turning either SHARED_INSTALL or READONLY_INSTALL when SUPPORT_WINDOWS_FRONTEND is on is also not supported and will cause cmake to exit with an error. To customize where the shared and read-only installations place files, pass -DCMAKE_INSTALL_PREFIX=prefix to install all the files within the given prefix (i.e. using -DCMAKE_INSTALL_PREFIX=/opt/Angband-4.2.3 would place all the files within /opt/Angband-4.2.3

or its subdirectories). For finer-grained placement of the files within the given prefix, you could also set CMAKE_INSTALL_BINDIR (for the subdirectory of prefix where the executable will be placed; by default that is bin), CMAKE_INSTALL_DATAROOTDIR (for the subdirectory of prefix to hold read-only data not configured for the site; by default that is share), CMAKE_INSTALL_SYSCONFDIR (for the subdrectory of prefix to hold data configured for the site; by default that is etc), and CMAKE_INSTALL_SHAREDSTATEDIR (for the subdirectory of prefix to hold writable persistent state; by default that is com). Because paths to the data are hardwired in the executable, setting the destination directory when running make (i.e. by setting DESTDIR) is not supported and will not work in general: set the destination when running cmake by setting the variables mentioned above.

Cross-building for Windows with Mingw

Many developers (as well as the auto-builder) build Angband for Windows using Mingw on Linux. This requires that the necessary Mingw packages are all installed.

This type of build now also uses autotools so the overall procedure is very similar to that for a native build. The key difference is setting up to cross-compile when running configure.

If your source files are from cloning the git repository, you'll first need to run this to create the configure script:

```
./autogen.sh
```

That is not necessary if your source files are from the source archive, a .tar.gz file, for a release.

Then configure the cross-comilation and perform the compilation itself:

```
./configure --enable-win --build=i686-pc-linux-gnu --hos
make install
```

You may need to change the –build and –host options there to match your system. Mingw installs commands like 'i686-w64-

mingw32-gcc'. The value of –host should be that same command with the '-gcc' removed. Instead of i686 you may see i686, amd64, etc. The value of –build should be the host you're building on (see http://www.gnu.org/savannah-checkouts/gnu/autoconf/manual/autoconf-2.68/html_node/Specifying-Target-Triplets.html#Specifying%20Names for gory details of how these triplets are arrived at). The 'make install' step only works with very recent version. For older ones, use this instead of the last step:

```
make
cp src/angband.exe .
cp src/win/dll/*.dll .
```

To run the result, you can use wine like this:

```
wine angband.exe
```

TODO: except for recent versions (after Angband 4.2.3) you likely need to manually disable curses (add –disable-curses to the options passed to configure), or the host curses installation will be found causing the build process to fail when linking angband.exe (the error message will likely be "cannot find -lncursesw" and "cannot find -ltinfo"). Most of the –with or –enable options for configure are not appropriate when using –enable-win. The ones that are okay are –with-private-dirs (on by default), –with-gamedata-in-lib (has no effect), –enable-release, –enable-more-gcc-warnings, and –enable-skip-old-int-typedefs.

Debug build

WARNING this build is intended primarily for debugging purposes. It might have a somewhat slower performance, higher memory requirements and panic saves don't always work (in case of a crash there is a higher chance of losing progress).

When debugging crashes it can be very useful to get more information about *what exactly* went wrong. There are many tools that can detect common issues and provide useful information. Two such tools that are best used together are AddressSanitizer (ASan) and UndefinedBehaviorSanitizer (UBSan). To use them you'll need to enable them when compiling angband:

```
./configure [options]
SANITIZE_FLAGS="-fsanitize=undefined -fsanitize=address"
```

Note that compiling with this tools will require installing additional dependancies: libubsan libasan (names of the packages might be different in your distribution).

There is probably a way to get these tools to work on Windows. If you know how, please add the information to this file.

Test cases

To compile and run the unit tests if you used ./configure –with-no-install, do this:

```
make tests
```

If you want to rerun just one part, say monster/attack, of the unit tests, that's most easily done by directly running from the top-level directory:

```
src/tests/monster/attack.exe
```

Previous versions put the test executables in src/tests/bin. With those versions, the line above would be:

```
src/tests/bin/monster/attack
```

There's a separate set of tests that use scripts to control a character in the full game. To run those tests, you'll need to enable the test module when running configure and then run the run-tests script in the top-level directory:

```
./configure --with-no-install --enable-test
make
./run-tests
```

To compile and run the unit tests and run the run-tests script while using CMake, do the following:

```
mkdir build && cd build
```

```
cmake -DSUPPORT_TEST_FRONTEND=ON ..
make alltests
```

If you only want the unit tests while using CMake, it's a little simpler:

```
mkdir build && cd build
cmake ..
make allunittests
```

Statistics build

If building directly for Linux/Unix using configure, you can get the statistics front end and support for the debugging commands related to statistics (see the descriptions for `S`, `D`, and `P` in Dungeon) by including –enable-stats in the options to configure. For that to work, you'll need to have sqlite3's headers and libraries installed (on Debian and Ubuntu, the libsqlite3-dev package and its dependencies provides those). If using CMake, pass -DSUPPORT_STATS_FRONTEND=ON to cmake to get the statistics front end and support for the debugging commnands related to statistics; like builds with configure that use –enable-stats, that requires sqlite3. With CMake, you also have an the option to only include support for the debugging commands related to statistics: pass -DSUPPORT_STATS_BACKEND=ON to cmake and either do nothing for SUPPORT_STATS_FRONTEND or explicitly turn it off by passing -DSUPPORT_STATS_FRONTEND=OFF to cmake.

When cross-compiling for Windows, the statistics front end is not useful (the Windows front end bypasses main.c and can not use the statistics front end). With configure, you could include support for debugging commands related to statistics by setting CFLAGS to include -DUSE_STATS:

```
./configure [your cross-compiling options] --enable-win
```

Windows

Using MinGW

This build now also uses autotools, so should be very similar to the Linux build. Open the MinGW shell (MSYS) by running msys.bat.

If your source files are from cloning the git repository, you'll first need to run this in the directory to create the configure script:

```
./autogen.sh
```

That is not necessary if your source files are from the source archive, a .tar.gz file, for a release.

Then run these commands:

```
./configure --enable-win
make install
```

The last step only works with very recent versions. For older ones, use "make" rather than "make install" and copy src/angband.exe, src/win/dll/libpng12.dll, and src/win/dll/zlib1.dll to the top-level directory.

Using Cygwin (with MinGW)

Use this option if you want to build a native Windows executable that can run with or without Cygwin.

Use the Cygwin setup.exe to install the mingw-gcc-core package and any dependencies suggested by the installer.

If your source files are from cloning the git repository, you'll first need to run this in the directory to create the configure script:

```
./autogen.sh
```

That is not necessary if your source files are from the source archive, a .tar.gz file, for a release.

Then run these commands:

```
./configure --enable-win --host=i686-pc-mingw32
make install
```

The last step only works with very recent versions. For older ones, use “make” rather than “make install” and copy src/angband.exe, src/win/dll/libpng12.dll, and src/win/zlib1.dll to the top-level directory.

If you want to build the Unix version of Angband that uses X11 or Curses and run it under Cygwin, then follow the native build instructions (./autogen.sh; ./configure; make; make install).

Using MSYS2 (with MinGW64)

Install the dependencies by:

```
pacman -S make mingw-w64-x86_64-toolchain mingw-w64-x86_
```

Additional dependencies for SDL2 client:

```
pacman -S mingw-w64-x86_64-SDL2 mingw-w64-x86_64-SDL2_im
        mingw-w64-x86_64-SDL2_ttf
```

Then run the following to compile with ncurses:

```
cd src
make -f Makefile.msys2
```

For SDL2, do:

```
cd src
make -f Makefile.msys2.sdl2
```

Very recent versions of Makefile.msys2.sdl2 allow use of SDL2 sound; to build with that you’ll need SDL2_mixer installed in addtion to the other SDL2 libraries mentioned above:

```
pacman -S mingw-w64-x86_64-SDL2_mixer
```

Then the executable with SDL2 sound support can be built with:

```
cd src
make -f Makefile.msys2.sdl2 SOUND=yes
```

Once built, go to the root of the source directory and start angband

by:

```
./angband.exe -uPLAYER
```

The ncurses client may not be able to start properly from msys2 shell, try:

```
start bash
```

and run:

```
export TERM=
./angband.exe -uPLAYER
```

Using eclipse (Indigo) on Windows (with MinGW)

- For eclipse with EGit, select File | Import..., Git | Projects from Git, Next >
- Clone your/the upstream repo, or Add your existing cloned repo, Next >
- Select "Use the New Projects Wizard", Finish
- In the New Project Wizard, select C/C++ | Makefile Project with Existing Code, Next >
- Give the project a name (Angband), * navigate to the repo you cloned in "Existing Code Location", * Select "C", but not "C++" * Choose "MinGW GCC" Toolchain, Finish
- Once the project is set up, r-click | Properties
- Go to C/C++ Build | Toolchain Editor, select "Gnu Make Builder" instead of "CDT Internal Builder"
- go to C/C++ Build, uncheck "Generate Makefiles automatically"

You still need to run ./autogen.sh, if your source files are from cloning the git repository, and ./configure manually, outside eclipse (see above)

Using Visual Studio

Blue Baron has detailed instructions for setting this up at:

src/win/angband_visual_studio_step_by_step.txt

Statistics build

The Windows front end bypasses main.c and can not use the statistics front end. It is possible to enable the debugging commands related to statistics (see the descriptions for `S`, `D`, and `P` in Dungeon). To do so, set your compiler options so that the USE_STATS preprocessor macro is set. When using mingw (either stand-alone or as part of Cygwin) and configure, include CFLAGS=-DUSE_STATS in the options to configure to do that.

Nintendo DS / Nintendo 3DS

Builds for the Nintendo DS are made using devkitARM and libnds (or libctru for the Nintendo 3DS respectively). All required dependencies can be installed by selecting the appropriate package group while following the installation instructions for devkitPro (https://devkitpro.org/wiki/Getting_Started).

The executable can then be built using:

```
cd src
make -f Makefile.nds
```

This will generate `angband.nds` in the current directory. For the Nintendo 3DS, replace the `Makefile.nds` part of the command with `Makefile.3ds`, and `angband.3dsx` will be generated instead.

Debugging

Homebrew can be debugged using a gdbstub-enabled emulator, such as a Windows Dev+ build of DeSmuMe (if you really dare to, note that it is very slow compared to real hardware) for the Nintendo DS or Citra for the Nintendo 3DS. A Nintendo 3DS that has been modified with custom firmware (such as Luma3DS) may also have the ability to debug homebrew on-device.

It is recommended to set/export `NDS_DEBUG=1` and to do a clean build when debugging, as this disables some optimization and enables more debugging information.

Once the GDB server has been set up (and the host and port noted), the GDB client can be loaded with the executable information:

```
/path/to/devkitARM/bin/arm-none-eabi-gdb angband.elf
```

The `angband.elf` file is a byproduct from the build process, and it has to match the executable that is currently running in the emulator or on the device. It is always named `angband.elf` for the Nintendo 3DS, and it's always either `angband.arm7.elf` or `angband.arm9.elf` for the Nintendo DS, depending on which processor should be debugged (as the main game runs on the ARM9 core exclusively, this will almost always be the core that should be debugged).

Once the GDB command prompt is available, the following command can be used to connect to the target device:

```
target remote <host>:<port>
```

Afterwards, the debugging target will pause automatically and it can be debugged as usual using GDB.

Cross-building for DOS with DJGPP

These instructions were written using a Slackware64-15.0 host.

Install the following cross-compiler: https://github.com/andrewwutw/build-djgpp

> git clone https://github.com/andrewwutw/build-djgpp.git cd build-djgpp DJGPP_PREFIX=$HOME/local/cross-djgpp ./build-djgpp.sh 10.3.0

Then build angband using the cross-compiler:

> cd angband/src PATH=$PATH:$HOME/local/cross-djgpp/bin make -f Makefile.ibm

Optionally build the documentation (requires Sphinx):

> make -f Makefile.ibm docs

To create the angband.zip distribution

```
make -f Makefile.ibm dist
```

Documentation

To convert the documentation from restructured text to the desired output format, you'll need Sphinx (https://www.sphinx-doc.org/en/master/) and, unless you change the theme in the documentation configuration, the sphinx-better-theme (https://pypi.org/project/sphinx-better-theme/ ; which can be installed via pip using:

```
pip install sphinx-better-theme
```

).

With those utilities in place and sphinx-build in your path, you can perform the conversion by running:

```
make html
```

in the docs subdirectory of the top-level directory in the source files. That will generate a _build/html directory with the result of the conversion; _build/html/index.html is the top-level help with links to everything else.

Other output formats besides HTML are possible. Run:

```
make
```

without any arguments in the docs subdirectory to see the formats that Sphinx can generate.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Debug Command Descriptions

Item Creation

Create an object `c`
: Provides a menu to let you create any object, and drops it on the floor.

Create an artifact `C`
: Provides a menu to let you create any artifact, and drops it on the floor.

Create a good object `g`
: Prompts for the number of objects to create and then creates that many good objects nearby.

Create a very good object `v`
: Prompts for the number of objects to create and then creates that many very good (“excellent”) objects nearby.

Play with an object `o`
: Lets you modify an object by randomly rerolling it as a normal, good, or excellent object or lets you edit its attributes, including quantity, ego type, presence or absence of curses, combat values, and modifiers. There’s also a statistics option to evaluate how likely worse, better, or matching objects of the same kind would be generated.

Test kind `V`
: Will prompt for a tval, as an integer. For that tval, creates one object of each sval and drops it nearby. There is a similar

option with the ‘c’ command, but this one will generate any instant artifacts associated with a tval and selects the tval by number rather than name.

Detection / Information

Detect all `d`
: Detects all traps, doors, stairs, treasure, and monsters nearby.

Magic Mapping `m`
: Maps the nearby dungeon.

Learn about objects `l`
: Makes you “aware” of all items with level less than or equal to 100.

Monster recall `r`
: Gives you full monster recall on all monsters or on a chosen monster.

Wipe recall `W`
: Resets monster recall on all monsters or on a chosen monster.

Unhide monsters `u`
: Reveals all monsters.

Wizard-light the level `w`
: Lights the entire level, as the Potion of Enlightenment.

Create spoilers `"`
: Lets you create a spoiler file for objects or monsters.

Teleportation

Teleport level `j`
: Allows you to teleport to any dungeon level instantly.

Phase Door `p`
: Teleports you up to 10 spaces away.

Teleport `t`
: Teleports you up to 100 spaces away.

Teleport to target `b`
: Teleports you to the targeted grid (or close to it, if it is occupied).

Character Improvement

Cure all maladies `a`
: Removes all curses, restores all stats, xp, hp, and sp, cures all bad effects, and satisfies your hunger.

Advance the character `A`
: Advances your character to level 50, maxes all stats, and gives you a million gold.

Edit character `e`
: Lets you specify your base stats, xp, and gold.

Increase experience `x`
: Prompts for an amount, up to 9999, to add to your current experience.

Rerate hitpoints `h`
: Rerates your hitpoints.

Monsters

Summon monster `n`
: Prompts you for the name or integer index of a monster, then summons that monster nearby.

Summon random monster `s`
: Prompts for a number and then summons that many random monsters near you.

Zap monsters `z`
: Prompts for a distance, up to the maximum sight range, and deletes all monsters within that distance.

Hit all in line of sight `H`
: Hits all monster in the line of sight for a large, 10000, amount of damage.

Dungeon

Create a trap `T`
: Prompts for the type of trap to create and places it on your square.

Perform an effect `E`
: Prompts for an effect type and its parameters. Then executes that effect.

Quit without saving `X`
: Quits the game without saving (prompts first).

Query the dungeon `q`
: Light up all the grids with a given square flag (see src/list-square-flags.h).

Query terrain `F`
: Light up all the grids with a given terrain type (see lib/gamedata/terrain.txt).

Collect stats `f` or `S`
: Collects stats on monsters and objects present on level generation. Requests number of runs, and whether diving or clearing levels, and outputs the results into the file 'stats.log' in the user directory. The comments in that file will be helpful for interpreting the results; for more in-depth information, it's best to check the implementation of stats_collect() in wiz-stats.c.

Collect disconnection stats `D`
: Generates several levels to collect statistics about how often all down staircases are inaccessible to the player, how often the player's starting location isn't valid, and how often a level has non-vault areas that are inaccessible to the player. The results are written to the message window, and maps of the

levels that are disconnected or have invalid starting locations are written to ‘disconnect.html’ in the user directory. Also collects general statistics about the layout of all the generated levels and writes them to ‘disconnect_gstat.txt’ in the user directory. For more in-depth details about what’s considered disconnected and what else is summarized about level generation, check the implementation for disconnect_stats() in wiz-stats.c.

Collect pit stats `P`
: Generates several pits of the room type you specify (pit, nest, or other) and computes a histogram of the types of monsters involved. The results are written to the message window.

Nick hack `_`
: Maps out the reachable grids (by the sound and scent algorithm) in successive distances from the player grid.

Push objects `>`
: Pushes objects off the targeted grid as a way of exercising push_object().

Write a map of the current level `M`
: Writes out a map of the current level as an HTML file.

Miscellaneous

Animations demo `G`
: Displays the graphics or characters used for animating projection effects.

Key log `L`
: Displays the recent keystrokes entered.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

How It Works

This document describes how Angband actually *works* at a high level. Individual sections referenced from the TOC are marked with anchors in square brackets to make grepping for them easier.

Contents

The Game

As you probably know if you're reading this, Angband is a roguelike game set in a high-fantasy universe. The game world is made up of levels, numbered from zero ("the town") to some maximum depth. Levels are increasingly dangerous the deeper they are into the dungeon. Levels are filled with monsters, traps, and objects. Monsters move and act on their own, traps react to creatures entering their square, and objects are inert unless used by a creature. The objective of the game is to find Morgoth at depth 100 and kill him.

Data Structures

There are three important top-level data structures in Angband: the 'chunk', the player, and the static data tables.

The Chunk

A chunk represents an area of dungeon, and contains everything inside it; this includes any monsters, objects, or traps inside the bounds of that chunk. A chunk also keeps a map of the terrain in its area. For unpleasant historical reasons, all monsters/objects/traps in a chunk are stored in arrays and usually referred to by index; each square of a chunk knows the indexes (if any) of monsters/objects/traps contained in it. A chunk also stores AI pathfinding data for its contained area. All data in the 'current' chunk is lost when leaving the level.

The Player

The player is a global object containing information about, well, the player. All the information in the player is level-independent. This structure contains stats, any current effects, hunger status, sex/race/class, the player's inventory, and a grab-bag of other information. Although there is a global player object, many functions instead take a player object explicitly to make them easier to test.

The Static Data

Angband's static data - player and monster races, object types, artifacts, et cetera - is loaded from the gamedata Files. Once loaded, this data is stored in global tables, sometimes referred to as the 'info arrays'. These arrays are generally declared in the header files of the code that uses them most, but they are mostly initialized by the edit-file code. The sizes of these arrays are stored in a 'maxima' structure, called z_info.

The Z Layer

The lowest-level code in Angband is the "Z" layer, which provides platform-independent abstractions and generic data structures. Currently, the Z layer provides:

Densely-packed bit flag arrays	`bitflag`
Colors	`color`
Debugging annotations	`debug`
Dice expressions	`dice`
Mathematical expressions	`math`
File I/O	`file`
String formatting	`format`
Rich messages	`msg`
Message buffering -lis	`msgt`
String interning	`string`
Queues	`queue`
Randomness	`random`
Sets	`set`
Wrapped text	`textblock`
Basic types	`types`

z-random.h	Random utility macros
z-virt.h	malloc() wrappers

Code in the Z layer may not depend on files outside the Z layer.

Key Abstractions

Certain game-specific abstractions are important and widely used in Angband to glue the UI code to the game engine. These are the command queue, which sends player commands to the game engine, and events, which indicate to the UI that the state of the game changed.

The command queue

TBD

Events

TBD

Files

Angband uses three types of files for storing data: gamedata files, which contain the game's static data, pref files, which contain UI settings, and save files, which contain the state of a game in progress.

Gamedata Files

Gamedata files use a line-oriented format where fields are separated by colons. The parser for this format is in `parser.h`. These files are mostly loaded at initialization time (see init.c - init_angband) and used to fill in the static data arrays (see The Static Data).

Pref Files

TBD

Savefiles

Currently, a savefile is a series of concatenated blocks. Each block has a name describing what type it is and a version tag. The version tag allows for old savefiles to be loaded, although the load/save code will only write new savefiles. Numbers in savefiles are stored in little-endian byte order and strings are stored null-terminated.

Control Flow

The flow of control through Angband is complicated and can be very non-obvious due to overuse of global variables as special-behavior hooks. That said, this section gives a high-level overview of the control flow of a game session.

Startup

Execution begins in main.c, which runs frontend-independent initialization code, then continues in the appropriate `main-*.c` file for the current frontend. After the game engine is initialized, the player is loaded (or generated) and gameplay begins.

`main.c` and `main-*.c`

main.c's `main()` is the entry point for Angband execution except on Windows, where main-win.c's `WinMain()` is used, on Nintendo DS, where a special `main()` in main-nds.c is used, and on OS X where main-cocoa.m's `main()` is used. The `main()` function is responsible for dropping permissions if Angband is running setuid, parsing command line arguments, then finding a frontend to use and initializing it. Once `main()` finds a frontend, it sets up signal handlers, sets up the display, and calls init.c - init_angband, which loads all the gamedata files and initializes other static data used by the game.

init.c - `init_angband`

The init_angband() function in init.c is responsible for loading and setting up static data needed by the game engine. Inside init.c, there

is a list of 'init modules' that have startup-time static data they need to initialize, these are registered in an array of module pointers in init.c, and init_angband() calls their initialization hooks before doing any other work. Finally it sets up the RNG.

ui-init.c - `textui_init`

The textui_init() function then loads the top-level pref file (see pref files), initializes the command queue (see the command queue), and configures subwindows.

ui-prefs.c - `process_pref_file`

The process_pref_file() function in ui-prefs.c is responsible for loading user pref files, which can live at multiple paths. User preference files override default preference files. See pref files for more details.

ui-game.c - `play_game`

This function calls start_game() to load a saved game if there is a valid save (see savefiles) or birth a new character if not. It then asks for a command from the player, and then runs the game main loop (see game-world.c - the game main loop), over and over until the character dies or the player quits

Gameplay

Once the simulation is set up, the game main loop in ui-game.c - play_game is responsible for stepping the simulation.

game-world.c - the game main loop

The main loop of the game, run_game_loop() is repeatedly called inside play_game(). Each iteration of the main loop is one "turn" in Angband parlance, or one step of the simulator. During each turn:

- All monsters with more energy than the player act
- The player acts

- All other monsters act
- The UI updates
- The world acts
- End-of-turn housekeeping is done

mon-move.c - process_monsters()

In Angband, creatures act in order of "energy", which roughly determines how many actions they can take per step through the simulation. The process_monsters() function in mon-move.c is responsible for walking through the list of all monsters in the current chunk (see the chunk) and having each monster act by calling process_monster(), which implements the highest level AI for monsters.

game-world.c - process_player()

The process_player() function allows the player to act repeatedly until they do something that uses energy. Commands like looking around or inscribing items do not use energy; movement, attacking, casting spells, using items, and so on do. The rule of thumb is that a command that does not alter game engine state does not use energy, because it does not represent an action the character in the simulation is doing. The guts of the process_player() function are actually handled by process_command() in cmd-core.c, which looks up commands in the game_cmds table in that file.

Keeping the UI up to date

Four related horribly-named functions in player-calcs.h are responsible for keeping the UI in sync with the simulated character's state:

- which deals with pack combining and dropping ignored items;
- which recalculates derived bonuses, AI data, vision, seen monsters, and other things based on the flags in `player->upkeep->update`;
- which signals the UI to redraw changed sections of the game state;
- which calls update_stuff() and redraw_stuff() if needed.

These functions are called during every game loop, after the player and all monsters have acted.

game-world.c - process_world()

The process_world() function only runs every 10 turns. It is responsible for the day/night transition in town, restocking the stores, generating new creatures over time, dealing poison/cut damage, applying hunger, regeneration, ticking down timed effects, consuming light fuel, and applying a litany of spell effects that happen 'at random' from the player's point of view.

Dungeon Generation

prepare_next_level() in generate.c controls the process of generating or loading a level. To signal that run_game_loop() in game-world.c should call prepare_next_level(), game logic calls dungeon_change_level() in player-util.c to set the necessary data in the player structure. When a level change happens by traversing a staircase, some other data in the player structure is set to indicate what should be done to connect stairs. That doesn't happen in dungeon_change_level() and is instead set directly, currently in do_cmd_go_up() and do_cmd_go_down() in cmd-cave.c.

With the default for non-persistent levels, loading only happens when returning to the town or when returning from a single combat arena. The code and global data for handling stored levels is in gen-chunk.c.

When a new level is needed, prepare_next_level() calls cave_generate(), also in generate.c. That initializes a global bit of state, a dun_data structure called dun declared in generate.h, for passing a lot of the details needed when generating a level. It then selects a level profile via choose_profile() in generate.c. The level profile controls the layout of the level. The available level profiles are those listed in list-dun-profiles.h and several aspects of each profile are configured at runtime from the contents of lib/gamedata/dungeon_profile.txt. With a profile selected, cave_generate() uses the profile's builder function pointer to attempt to layout the new level. Those function pointers are

initialized when list-dun-profiles.h is included in generate.c. The level layout functions all have names with the name of the profile followed by *_gen*, classic_gen() for classic levels as an example. Those functions are defined in gen-cave.c.

Three of the level layout functions, classic_gen(), modified_gen(), and moria_gen() follow the same basic procedure. They divide the level into a grid of rectangular blocks where, in general, each block can only contain one room though a room could occupy many blocks. They then try to randomly place rooms in those blocks until some criteria is met. Room selection is configurable from lib/gamedata/dungeon_profile.txt and uses the predefined room types listed in list-rooms.h. When building a room, those level layout functions use the convenience function, room_build() from gen-room.c. That, in turn, calls the appropriate function to build the type of room chosen. The names of the room building functions have *build_* followed by the name of the room type, build_simple() for instance. Those functions are defined in gen-room.c. Once the rooms are built, there's an initial pass to connect them with corridors. That happens in gen-cave.c's do_traditional_tunneling(). A second pass, to try and ensure connectedness though vault areas can disrupt that, is then done with ensure_connectedness(). At that point, most other features (mineral veins, staircases, objects, and monsters) are added. Some features will have already been added through some of the types of rooms.

The other layout functions are more of a grab bag. They are all in gen-cave.c. Many of them have portions that are caverns or labyrinths. Those are generated using cavern_chunk() or labyrinth_chunk(), respectively, in gen-cave.c.

Monster AI

TBD

Stats

The stats generation code aims to make it easy to analyze object generation, monster generation, and other Angband processes suitable for Monte Carlo simulation. The stats pseudo-visual module

will repeatedly create a character, walk her down the dungeon, and, for each dungeon level, kill the monsters there and dump information about the monsters and objects. The end result is a SQLite3 database, written to the stats subdirectory of Angband's user directory. A similar procedure is used by the `S` debugging command. It will generate a text file summarizing the monsters and objects generated. That output may be more accessible, since one doesn't have to deal with the structure of the database, but the database stores finer-grained classifications of the objects and monsters.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Documentation on documentation

Documentation is now written in ReStructuredText, a lightweight and readable markup language designed for Python.

Conventions

ReStructuredText leaves much freedom in the syntax - in our help files we try to stick to consistent conventions as in the following.

Markup for sectioning

```
==============
Document title
==============

Section title
=============

Subsection title
----------------

Sub-subsection title
********************
```

Blank and spaces

One blank line before and after every paragraph. Two-char indent in definition lists.

Inline help parser syntax

As to the time when this documentation was last updated, the online help parser programmed into Angband does the following:

- ignore colons (`|`)
- ignore paragraphs starting with `..` (including their trailing blank line)
- recognize link targets (`.. _something`) and be able to open a file to the right position using them.
- recognize "menu" links for the online help navigation. Links are given with the following format (`g` is the hotkey here):

```
.. menu:: [g] option.txt
```

Tricks

Sometimes it is tricky to get a thing to display in the correct way in both the online help and the compiled RST. This is often due to the complex interaction between backslashes and backticks in RST. In particular, in several places we need to use in RST a syntax such as

```
``D``\isarm
```

If this appeared directly in the help file, then the online help parser would display a spurious backslash character `\` before the `i` letter. It would be quite complicated to instruct it to parse backslashes using the full RST syntax. Instead, we adopt the following workaround: we write

```
this is a paragraph containing the word |``D``isarm| jus
```

and later in the file we define a substitution directive

```
.. |``D``isarm| replace:: ``D``\isarm
```

Magically this works, since the online help parser *skips* pipes.

v4.2.4-240-g8f823a77-dirty [image]

The Angband Manual

Index

Table of Contents

Landmarks